Tampereen ammattikorkeakoulu

Tiedonsiirtokaapeleiden testausjärjestelmän suunnittelu ja toteutus

Jini Autio

OPINNÄYTETYÖ
Kesäkuu 2020

Sähkö- ja automaatiotekniikka
Älykkäät koneet

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Sähkö- ja automaatiotekniikka
Älykkäät koneet

AUTIO, JINI:
Tiedonsiirtokaapeleiden testausjärjestelmän suunnittelu ja toteutus

Opinnäytetyö 52 sivua, joista liitteitä 0 sivua
Kesäkuu 2020

Opinnäytetyön tarkoituksena oli kehittää tiedonsiirtokaapeleiden testausjärjestelmä nopeuttamaan ja helpottamaan monijohtimisten tiedonsiirtokaapeleiden toimintakunnon toteamista. Työ tehtiin Millog Oy:lle Lylyn toimipaikkaan, ja sen tavoitteena oli onnistuneesti automatisoida nykyisin käsin tehtävä testausprosessi. Opinnäytetyössä käsitellään koko tuotekehitysprosessi ideasta aina valmiin tuotteen testaamiseen asti. Työ sisältää elektroniikka-, piirilevy-, ja ohjelmistosuunnittelua sekä 3D-mallinnusta ja -tulostusta.

Opinnäytetyössä päädyttiin innovatiiviseen ratkaisuun, jossa aikaisemmin asentajan tekemä työ korvattiin mikrokontrollerilla. Mikrokontrollerin GPIO-nastojen riittämättömyys haluttuun toiminnallisuuteen ratkaistiin hyödyntämällä multiplekssereitä ja siirtorekistereitä. Suunniteltu järjestelmän mittausperiaate perustui testisignaalin syöttämiseen kaapelin toisesta päästä ja kaapelissa tapahtuvan jännitehäviön mittaamiseen sen toisesta päästä käyttämällä mikrokontrolleria. Järjestelmään suunniteltiin helppokäyttöinen käyttöliittymä, jossa kaikkea toiminnallisuutta ohjataan painonapin ja LCD-näytön avulla. Mikrokontrollerin koodi kirjoitettiin Arduino IDE-ohjelmistolla, ja sitä kertyi kaiken kaikkiaan noin 1850 riviä. Järjestelmälle suunniteltiin ja tilattiin piirilevy koekytkentäalustalle rakennetun prototyypin perusteella. Lopuksi järjestelmän kotelot valmistettiin 3D-mallintamalla ja -tulostamalla.

Lopputuloksena saatiin aikaan tiedonsiirtokaapeleiden testausjärjestelmä, joka pystyy toteamaan jopa 48-johtimisten tiedonsiirtokaapeleiden toimintakunnon luotettavasti ja nopeasti. Järjestelmästä tuli erittäin helppokäyttöinen, ja siihen saatiin sisällytettyä paljon hyvää toiminnallisuutta. Järjestelmään jäi kuitenkin vielä paranneltavaa ja kehitettävää. Mikrokontrolleria vaihtamalla järjestelmän mittatarkkuutta voitaisiin parantaa huomattavasti ja ohjelmastakin saataisiin paljon yksinkertaisempi. Lisäksi järjestelmän häiriösuojausta voitaisiin parannella, jolloin myös toimintavarmuus paranisi entisestään.

Asiasanat: Tiedonsiirtokaapeleiden testausjärjestelmä, ohjelmointi, piirilevyn suunnittelu, 3D-mallinnus, 3D-tulostus

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Electrical Engineering
Intelligent Machines

JINI AUTIO:
Design and implementation of a Data Transfer Cable Testing System

Bachelor's thesis 52 pages, appendices 0 pages
June 2020

The purpose of this thesis was to develop a testing system for data transfer cables in order to speed up and to simplify testing of the operational condition of multi-wire data transfer cables. This was achieved by automating a process which was previously done by hand. The work was done for Millog Oy's Lyly office. This thesis goes briefly through the whole product development process, from the idea to the testing of the final product. The work includes the design of electronics, circuit board and software, as well as 3D-modeling and 3D-printing.

This thesis describes the development and implementation of an innovative solution, thanks to which the work previously done by an electronic assembler can be done by a microcontroller. During the design process, GPIO pins proved to be an unfitting choice for realization of all the desired functionalities in the system. This was solved by implementing multiplexers and shift registers. Cable condition measurement was implemented by feeding a test signal from one end of the cable and measuring the voltage drop at the other end with a microcontroller. A simple user interface was then created, where all functionality of the system is controlled with a push button and an LCD-display. The program for the microcontroller was created with Arduino IDE software and it included approximately 1850 lines of code. Then, a printed circuit board based on a prototype made on a test board was designed and ordered for the system Finally, casing was created for the system by 3D-modeling and 3D-printing.

This thesis resulted in data transfer cable testing system that is able to determine reliably and quickly the operational condition of up to 48 wire data transfer cables. The system is very easy to use and includes many useful functionalities. However, there is still room for improvements and development. By changing the microcontroller, the accuracy of the system could be significantly improved. That way also the program could be simplified. In addition, the interference protection of the system could be improved, which would also further improve operational reliability.

Key words: data transfer cable testing system, programming, PCB-designing, 3D-modeling, 3D-printing

SISÄLLYS

LYHENTEET JA TERMIT

MUX	Multiplexer, Multiplekseri
LCD	Liquid Crystal Display, Nestekidenäyttö
IDE	Integrated Developement Environment, Ohjelmointiympäristö
GPIO-nasta	General Purpose Input Output, Yleismallinen I/O-nasta
PWM	Pulse Width Modulation, Pulssinleveysmodulaatio
A/D-munnin	Analog to Digital Converter, Analogia-Digitaalimuunnin

1 JOHDANTO

Nykyään lähes kaikkiin laitteisiin ja järjestelmiin on integroitu jonkun verran älyä. Tästä syystä erilaiset tiedonsiirtomenetelmät yleistyvät ja kehittyvät jatkuvasti. Suurin osa tiedonsiirrosta tapahtuu kuitenkin edelleen erilaisia tiedonsiirtokaapeleita pitkin, sillä ne ovat toimintavarma ja kustannustehokas tapa siirtää tietoa erilaisten laitteiden ja järjestelmien välillä.

Sama laite tai järjestelmä voi usein vaatia monia erilaisia kaapeleita toimiakseen. Kun huomioidaan, että tällaisten kaapeleiden testaaminen suoritetaan edelleen lähes poikkeuksetta manuaalisesti yleismittarilla mittaamalla, niin yhdenkin tällaisen monilinjaisen tiedonsiirtokaapelin toimintakunnon toteaminen ja esimerkiksi siitä vian löytäminen on aikaa vievä ja tarkkuutta vaativa tehtävä jopa koulutetulle ammattilaiselle.

Tämä opinnäytetyö käsittelee monilinjaisten tiedonsiirtokaapeleiden testausjärjestelmän kehitysprojektia. Työn tarkoituksena on mahdollistaa erityyppisten tiedonsiirtokaapeleiden testaaminen luotettavasti ja nopeasti. Opinnäytetyön tavoitteena on onnistuneesti automatisoida tiedonsiirtokaapeleiden testaaminen ja näin ollen säästää kaapeleiden testaamiseen kuluvia resursseja.

Opinnäytetyö tehtiin Millog Oy:lle Lylyn toimipisteeseen, jonka vastuualueeseen kuuluu Suomen Puolustusvoimien viestimateriaalin huolto ja kunnossapito. Tämän työn tuloksena syntyi tiedonsiirtokaapeleiden testausjärjestelmä Millog Oy:lle Lylyn toimipisteeseen ja tätä järjestelmää tullaan käyttämään Suomen Puolustusvoimien viestimateriaalin kaapeleiden tarkastamisen sekä vikakorjauksen nopeuttamiseen ja helpottamiseen.

2 YRITYSESITTELY

2.1 Yleistä

Tämä opinnäytetyö tehtiin Millog Oy:lle Lylyn toimipaikkaan, joka sijaitsee Juupajoella. Millog Oy on vuonna 2006 perustettu moniteknisen kaluston kunnossapitoon, materiaalipalveluihin ja elinjakson hallintaan erikoistunut suomalainen yhtiö, jonka pääkonttori sijaitsee Tampereella. Millog Oy:n pääomistajat ovat Patria Oyj ja Insta Group Oy, joista Patrialla on enemmistöomistus. (Millog Oy 2018.)

Yhtiön toiminta perustuu pitkäaikaisiin kumppanuuksiin. Suomen puolustusvoimien strategisena kumppanina, Millog Oy ylläpitää maa- ja merivoimien kalustoa sekä myös erikseen sovitulta osalta ilmavoimien kalustoa. Yhtiön liikevaihto vuonna 2019 oli 214,5 miljoonaa, josta tilikauden voitto oli 16 miljoonaa. Vuonna 2019 Millog Oy työllisti 1056 henkilöä 22 toimipaikalla (kuvio 1). (Asiakastieto Oy 2020.)

KUVIO 1. Millog Oy:n toimipaikat kartalla (Millog Oy 2018 muokattu 13.5.2020)

2.2 Lylyn toimipaikka

Lylyn toimipaikka toimii vanhalla Suomen Puolustusvoimien viestivarikolla Juupajoella sijaitsevassa pienessä Lyly nimisessä kylässä. Toimipaikka siirtyi Puolustusvoimilta Millog Oy:n omistukseen vuonna 2009 ja muutti nimensä Lylyn viestivarikosta Lylyn toimipaikaksi. Toimipaikan pääasiallisena vastuualueena on Puolustusvoimien viestimateriaalin huolto ja kunnossapito. Huoltoon ja kunnossapitoon kuuluvaa materiaalia ovat muun muassa kenttäviestimateriaali eli kenttäpuhelin- ja sanomalaitekalusto sekä radiokalusto. Lisäksi myös antennimastot ja sähkövoimakoneet huolletaan Lylyn toimipaikassa. Toimipaikalla tehdään myös paljon ajoneuvo- ja sähkökonttivarusteluita sekä myös erilaisten energian varastointijärjestelmien huoltoa ja kunnossapitoa. Tällaista materiaalia ovat muun muassa: akut, akkuvaraajat ja UPS-järjestelmät. (Millog Oy 2018.)

3 KAAPELEIDEN TESTAAMINEN JA SEN HAASTEET

Monilinjaisten tiedonsiirtokaapeleiden kytkennän oikeellisuuden toteaminen on aikaa vievä ja virhealtis prosessi. Yleisin tapa testata kaapeleiden kytkentä on mitata johdinten jatkuvuus käyttämällä yleismittaria. Yleismittarilla käydään läpi kaikki mahdolliset liittimien sisäiset ja liittimien väliset yhteydet. Yleismittarilla mittaaminen on perehdytetylle asentajalle yksinkertaista ja helppoa, mutta kun mitataan kaapeleita, joissa on useita johtimia ja erilaisia sisäisiä kytkentöjä, prosessi monimutkaistuu huomattavasti ja muuttuu virhealttiimmaksi. Sen lisäksi, että kaapelin testaajan tulee osata käyttää yleismittaria, tarvitsee hänen myös muistaa tai kirjata ylös mittaamansa kaapelin liitinten jokaisen eri nastan välinen resistanssi ja joko verrata tuloksia kaapelin oikeelliseen kytkentään heti tai lopuksi, kun kaikki mahdolliset yhteydet on mitattu ja kirjattu. Pienikin virhe yhdenkin yhteyden tai katkoksen toteamisessa saattaa aiheuttaa ehjän kaapelin hylkäämisen tai viallisen kaapelin toteamisen ehjäksi.

Tiedonsiirtokaapeleita löytyy kaikkialta. Kaapeleissa käytetään erilaisia liittimiä ja kaapeleiden johdinmäärät voivat vaihdella muutamasta johtimesta useampaan kymmeneen johtimeen. Lisäksi jokaisella tiedonsiirtokaapelilla on yleensä oma tarkasti määritelty käyttötarkoituksensa ja sen mukaan määräytyvä sisäinen kytkentä. Eli vaikka kahdessa kaapelissa olisikin samat liittimet niin niiden sisäinen kytkentä saattaa erota huomattavasti. Tällaisten kaapeleiden sekoittamisella keskenään voi olla vakavia seurauksia.

Kaapeleiden kytkennän mittaaminen oikeaoppisesti vie paljon aikaa ja mitä monijohtimisempi kaapeli on, sitä enemmän sen testaamiseen joudutaan käyttämään arvokkaita työtunteja. Lisäksi mittausprosessin virhealttiuden takia, usein joudutaan vielä suorittamaan uudelleenmittaus, sillä pienikin ajatuskatkos mittauksen aikana altistaa mittauksen inhimillisille virheille, jolloin testiin ei voida enää luottaa. Pahimmassa tapauksessa pieni mittausvirhe kaapelia mitattaessa, saattaa aiheuttaa satojen, jopa tuhansien eurojen laitteen tai laitteiston vaurioitumisen.

4 TEKNINEN MÄÄRITTELY

Testausjärjestelmän suunnittelun lähtökohtana oli sille asetetut vaatimukset. Järjestelmän tuli kyetä testaamaan kaapeli ja ilmoittamaan sen toimintakunto luotettavasti ja nopeasti. Lisäksi sen tuli soveltua useille eri kaapelityypeille ja riittävän monijohtimisille kaapeleille. Kokemuksesta tiedettiin, että monijohtimisin tiedonsiirtokaapeli, jota käytetään ja tarkistetaan Millog Oy:n Lylyn toimipaikassa, on 32-johtiminen. Tämän lisäksi oli huomioitava, että kaapelin ympärillä olevan suojasukan jatkuvuus piti myös tarkistaa. Näin saatiin minimimäärä johtimia, joiden mittaamiseen valmiin testausjärjestelmän on kyettävä, jotta se palvelisi Millog Oy:n tarpeita. Testausjärjestelmän tulisi siis kyetä mittaamaan luotettavasti vähintään 33-johtimisia kaapeleita ja ilmoittamaan niiden toimintakunto luotettavasti ja nopeasti.

Järjestelmän haluttiin myös olevan helppokäyttöinen, jotta sen käyttöönotto onnistuisi sujuvasti, sillä muutosvastaisuus isoissa yhtiöissä on välillä todellinen ongelma. Vaikka toimintaperiaate saikin olla monimutkainen, haluttiin käyttöliittymästä selkeä ja yksinkertainen. Lisäksi järjestelmän tulisi olla kompaktin kokoinen, eikä sen virittämisessä käyttökuntoon tulisi kestää kauan. Tarvittaessa järjestelmän tulisi myös olla monistettavissa suhteellisen pienellä vaivalla.

5 KOMPONENTIT

5.1 Mikrokontrollerialustat

Mikrokontrollerialusta on piirikortti, jossa on mikrokontrolleri, ja kaikki sen toiminnan kannalta vaadittavat komponentit, jotta sitä voidaan helposti alkaa ohjelmoida käyttäen yhteensopivaa IDE-ohjelmistoa (Integrated Developement Environment). Mikrokontrollerin nastat ja liittimet on tuotu alustan reunoille, jotta siihen on helppo kytkeä ulkoisia laitteita kuten sensoreita, kytkimiä ja erilaisia indikaattoreita.

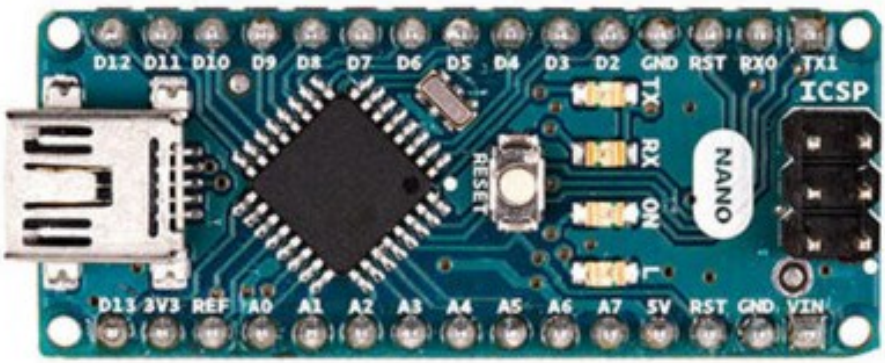
Alustoja on monenlaisia ja yleensä mikrokontrollereiden valmistajilla on omansa, sekä IDE-ohjelmistonsa. Tällä hetkellä tunnetuimpia ovat erilaiset avoimeen lähdetietoon perustuvat Arduinon mikrokontrollerialustat. Helpon IDE-ohjelmistonsa ja hyvän dokumentaationsa takia Arduino mikrokontrollerialustat mahdollistavat matalan kynnyksen oppia mikrokontrollereiden toimintaa ja kehittää monimutkaisiakin sovelluksia.

Mikrokontrolleri on pieni tietokone, josta löytyy prosessori, erityyppisiä muisteja ja GPIO-nastoja (General Purpose Input Output) ulkoisten laitteiden ohjaamiseen. Mikrokontrollereilla halutaan myös usein lukea analogista jännitettä. Tähän tarkoitukseen mikrokontrollerissa voi olla A/D-muunnin. A/D-muunnin muuntaa analogisen jännitteen digitaaliseksi muuttujan arvoksi, jolloin tätä arvoa voidaan käsitellä ohjelmallisesti. Useat mikrokontrollerit pystyvät myös tuottamaan PWM-signaalia. PWM-signaalin tehollista jännitettä voidaan muuttaa muuttamalla signaalin pulssisuhdetta ohjelmallisesti. Mikrokontrollereissa on myös erilaisia ohjelmoitavia ajastimia, jotka mahdollistavat kontrollerin toiminnan ajoittamisen ohjelmallisesti. Lisäksi mikrokontrollereihin on sisäänrakennettu erilaisia tiedonsiirtoväyliä, jotka mahdollistavat sen liitettävyyden muihin laitteistoihin. Tällaisia tiedonsiirtoväyliä voivat olla esimerkiksi I²C, SPI ja USART.

5.1.1 Arduino Nano V3.0

Arduino Nano on mikrokontrollerialusta, jota ohjelmoidaan Arduino IDE-ohjelmistolla (kuva 1). Nanossa on valmiiksi käynnistyksenlataaja, joten se voidaan ohjelmoida suoraan kehitysalustassa olevasta USB-portista. Siinä on 8 analogista sisäänmenoa ja 14 digitaalista GPIO-nastaa, joista 6 nastaa pystyy antamaan ulos pulssinleveysmoduloitua signaalia. Tämä mikrokontrollerialusta voi käyttää käyttöjännitteenään joko 7-20 voltin jännitettä syötettynä VIN-nastaan tai 5 voltin jännitettä syötettynä suoraan 5V-nastaan. Alusta saa käyttöjännitteensä tarvittaessa myös USB-portin kautta. (Farnell 2020.)

Mikrokontrollerina Nanossa on ATmega328P. ATmega328P on suorituskykyinen, mutta pienitehoinen, 8-bittinen mikrokontrolleri, jonka kellotaajuus on 16 MHz. ATmega328P käyttää ohjelmamuistinaan Flash-muistia, jota tässä mikrokontrollerissa on 32 kB. Ohjelmamuisti on muistia, johon mikrokontrolleria ohjaava ohjelma tallennetaan. Työmuistina tässä mikrokontrollerissa on SRAM-muisti (Static Random Acces Memory). Työmuistin määrä tässä mikrokontrollerissa on 2 kB. Työmuisti on muistia, johon mikrokontrolleri tallentaa ohjelman suorituksen aikaiset muuttujat. (Farnell 2020.)

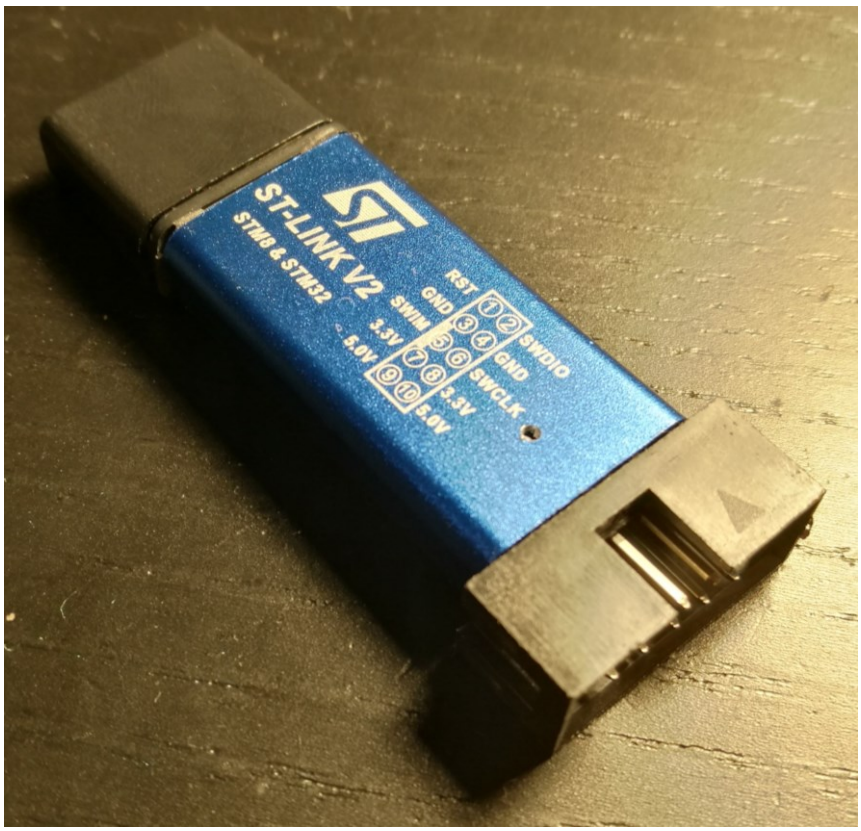


KUVA 1. Arduino Nano mikrokontrollerialusta (Arduino 2020)

5.1.2 STM32F103C8T6 (”Bluepill”)

STM32F103C8T6, tunnetaan myös nimellä Bluepill, on 32-bittinen mikrokontrollerialusta, jota voidaan ohjelmoida usealla eri IDE-ohjelmistolla. Tunnetuksi tämän mikrokontrollerialustan on tehnyt sen yhteensopivuus Arduinon IDE:n kanssa. Siitä onkin tullut Arduinon mikrokontrollerialustoille kova kilpailija, sillä Bluepill on halpa ja laskentateholtaan sekä muistiltaan ylivoimainen lähes kaikkiin Arduinon alustoihin nähden.

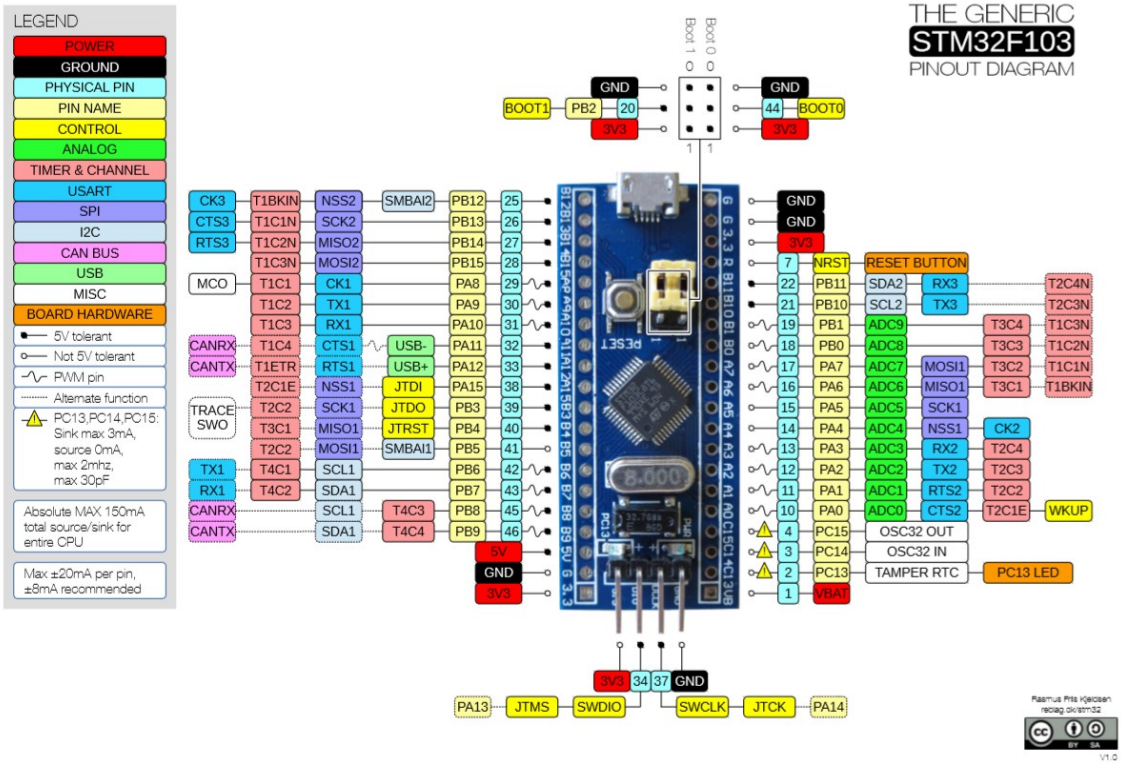
Ainoa merkittävä huono puoli on, että Bluepill-mikrokontrollerialustassa ei ole valmiiksi asennettuna käynnistyksenlataajaa, joka mahdollistaisi USB-portin kautta ohjelmoinnin. Tästä syystä sen ohjelmoimiseen tarvitaan erillinen ohjelmoija kuten ST-LINK V2 (kuva 2). Tähän mikrokontrollerialustaan on kuitenkin mahdollista ladata käynnistyksenlataaja ulkoisella ohjelmoijalla, jolloin myös USB-portin kautta ohjelmoiminen tulee mahdolliseksi. Kuvassa 2, mustan suojahatun alla on USB-liitin ja toisessa päässä on sarjaliitin mikrokontrollerin liittämiseksi ohjelmoijaan.



KUVA 2. ST-LINK V2 ohjelmoija.

Bluepill-mikrokontrollerialusta voi käyttää käyttöjännitteenään 3,3 voltin jännitettä syötettynä suoraan 3,3V-nastaan tai 5 voltin jännitettä syötettynä 5V-nastaan, sekä USB-liittimen kautta tulevaa 5 voltin jännitettä. Kehitysalustassa on 3,3 voltin regulaattori, joka on kytkettynä alustassa olevaan 5V-nastaan ja USB-portin jännitesyöttöön. Sisäinen regulaattori on kuitenkin sen verran pienitehoinen, että sen kautta ei ole suositeltavaa kuljettaa suuria tehoja.

Mikrokontrollerina tässä alustassa on 32-bittinen STM32F103C8T6 ARM Cortex-M3, jonka kellotaajuus on 72 MHz. Tässä mikrokontrollerissa on 37 GPIO-nastaa joista 10 nastaa voi toimia analogisena sisäänmenona ja 15 nastaa pystyy syöttämään pulssinleveysmoduloitua signaalia (kuvio 2). Kontrollerin jännitetaso on 3,3 volttia, mutta suuri osa nastoista on jännitekestoltaan 5 volttia. Lisäksi mikrokontrollerissa on runsaasti erilaisia väyliä sen liitettävyyden mahdollistamiseksi. Mikrokontrollerista löytyy muun muassa kaksi SPI-väylää, kaksi I²C-väylää, kolme USART-väylää ja jopa CAN-väylä (kuvio 2). Kaiken tämän lisäksi jokaiseen GPIO-nastaan voidaan myös kytkeä ohjelmallisesti joko ylös- tai alasvetovastus. (STMicroelectronics 2020.)

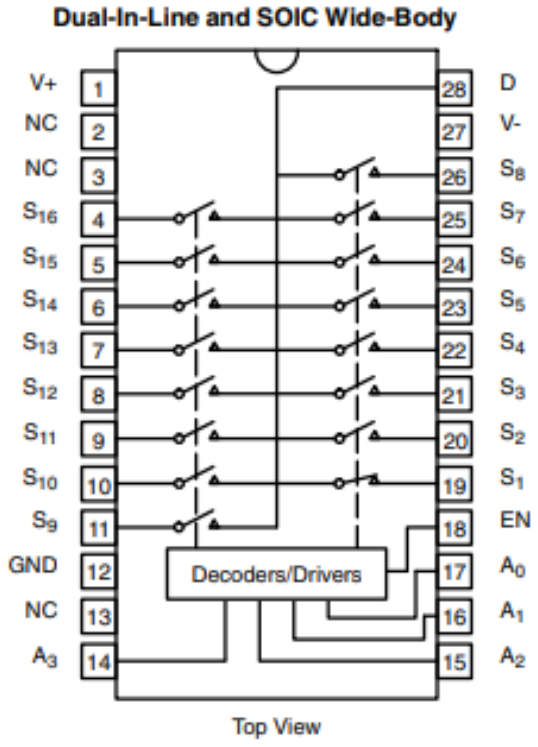


KUVIO 2. STM32F103C8T6 mikrokontrollerialustan nastat ja niihin kytketyt toiminnallisuudet (Techshopbd 2013)

5.2 Multiplekserit

Multiplekseri on elektroniikan komponentti, joka toimii kuin digitaalisesti ohjattava vaihtokytkin (kuvio 3). Multipleksereitä on useita eri variaatioita, joissa sisäisten kytkimien määrät voivat vaihdella muutamasta useaan kymmeneen. Myös multipleksereiden sisäisistä kytkennöistä on olemassa erilaisia variaatioita, mutta yleisin kytkentä on sellainen, jossa sisäisillä kytkimillä on yksi yhteinen nasta ja kytkimien toiset päät on tuotu omille nastoilleen (kuvio 3).

Kun multiplekserille annetaan digitaalinen ohjaus sen ohjausnastoihin, se kytkee ohjauksella valitun kytkimen kiinni yhteiseen nastaan. Signaali voi kulkea näiden multiplekserin sisäisten kytkimien kautta molempiin suuntiin. Tämä toiminnallisuus mahdollistaa yhden ja saman linjan kytkemisen useampaan eri linjaan tai toisin päin. Multipleksereiden sisäisillä kytkimillä on myös huomattavasti suurempi impedanssi, kuin esimerkiksi tavallisilla mekaanisilla kytkimillä. Tämä tulee ottaa huomioon multiplekseriä valittaessa, sillä eri multipleksereiden sisäisten kytkimien impedanssi voi vaihdella kymmenistä satoihin ohmeihin ja se on yleensä ilmoitettu multiplekserin datalehdessä.



KUVIO 3. DG406 multiplekserin sisäinen kytkentä ja nastajärjestys (Vishay Siliconix, s. 1)

Ohjausnastojen määrä määräytyy multiplekserin koon eli sisäisten kytkimien määrän mukaan. Esimerkiksi 16-kanavainen multiplekseri tarvitsee neljä ohjausbittiä, koska neljällä bitillä saadaan aikaan kuusitoista erilaista kombinaatiota, ja kukin kombinaatio yhdistää multiplekserin yhteisen nastan johonkin sen linjoista. Yleensä multipleksereissä on myös enable-nasta, jolla piiri voidaan kytkeä päälle ja pois päältä. Kun enable-nastaan syötetään digitaalinen ohjaus, se nostaa multiplekserin kytkinlähdöt korkeaimpedanssiseen tilaan. Taulukosta 1 nähdään miten DG406 analogisen multiplekserin ohjaaminen tapahtuu. Jos DG406 multiplekserin enable-nasta asetetaan nollaan volttiin, niin multiplekserin lähdöt asettuvat korkeaimpedanssiseen tilaan. Kun enable-nastaan syötetään viiden voltin jännite ja ohjausnastoihin ei ole syötetty signaalia niin multiplekserin yhteinen nasta kytkeytyy sen ensimmäiseen kytkinnastaan. Loput kytkimen asennot saadaan käytyä läpi syöttämällä digitaalinen ohjaus ohjausnastoihin A0 - A3 (taulukko 1).

TAULUKKO 1. DG406 multiplekserin ohjaustaulukko (Vishay Silconix, s. 2).

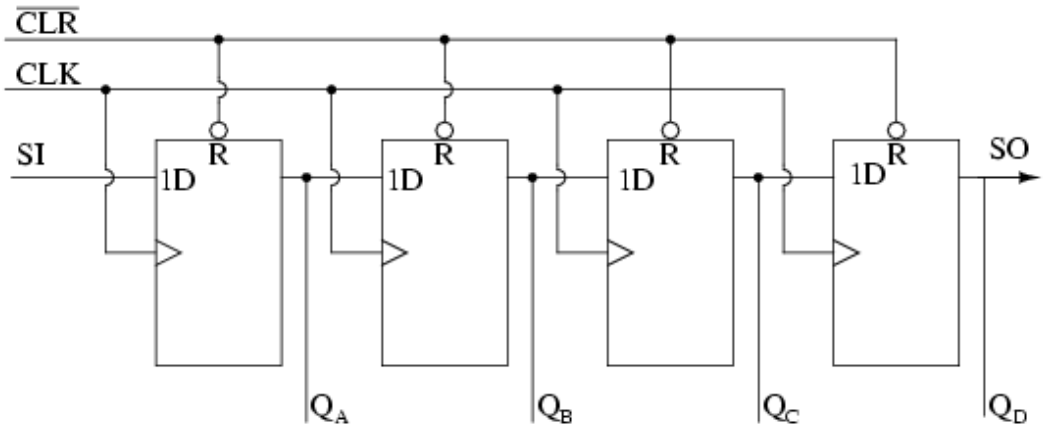
TRUTH TABLE (DG406)					
A_3	A_2	A_1	A_0	EN	ON SWITCH
X	X	X	X	0	None
0	0	0	0	1	1
0	0	0	1	1	2
0	0	1	0	1	3
0	0	1	1	1	4
0	1	0	0	1	5
0	1	0	1	1	6
0	1	1	0	1	7
0	1	1	1	1	8
1	0	0	0	1	9
1	0	0	1	1	10
1	0	1	0	1	11
1	0	1	1	1	12
1	1	0	0	1	13
1	1	0	1	1	14
1	1	1	0	1	15
1	1	1	1	1	16

Enable nasta mahdollistaa esimerkiksi sen, että yhden tai useamman multiplekserin ohjaukset ja yhteiset nastat voidaan kytkeä yhteen, ja erikseen annettavilla enable-signaaleilla voidaan valita, mikä multiplekseri hyödyntää annetun ohjauksen ja asettuu haluttuun asentoon. Näin multipleksereitä voidaan kytkeä rinnakkain tarvittava määrä.

5.3 Siirtorekisterit

Siirtorekisteri on elektroniikan komponentti, joka nimensä mukaisesti siirtää rekisteriin syötettyä dataa bitti kerrallaan annetun kellopulssin tahdissa. Siirtorekistereiden sisältä löytyy joukko ketjuun kytkettyjä kiikkuja, jotka pystyvät siirtämään vain yhtä bittiä kerrallaan. Kun kiikun toiseen sisäänmenoon syötetään bitti ja toiseen kellopulssi, kiikku siirtää datalinjaan syötetyn bitin lähtöönsä, joka on samalla seuraavan kiikun sisäänmeno. Näin jokaisen kellopulssin aikana data siirtyy yhden bitin kerrallaan eteenpäin.

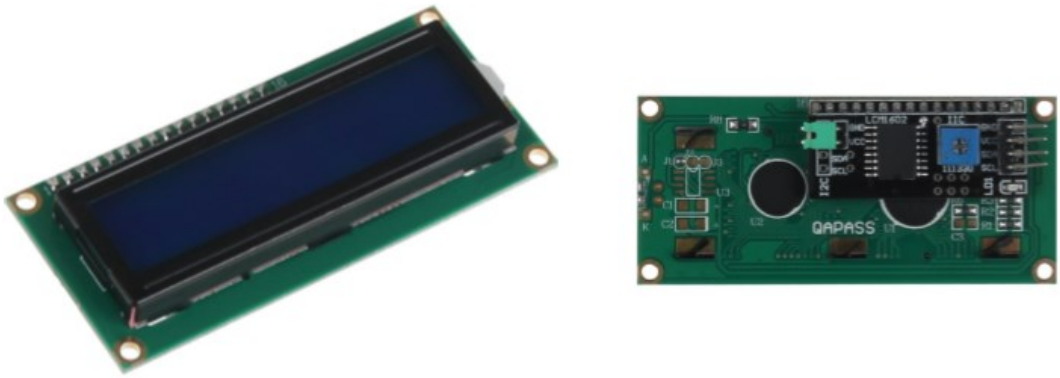
Siirtorekistereitä on neljää eri tyyppiä SISO (Serial In - Serial Out), SIPO (Serial In - Parralel Out) (kuvio 4.), PISO (Parralel In - Serial Out) ja PIPO (Parralel In - Parralel Out). Käytetyimmät kaksi rekisterityyppiä ovat SISO ja SIPO. Siirtorekistereitä käytetään datan prosessointiin. Esimerkiksi SIPO-tyyppistä siirtorekisteriä käytetään usein sarja/rinnan muunnoksen tekemiseen (kuvio 4). Siirtorekistereitä on myös helppoa kytkeä sarjaan, sillä yleensä sen viimeisen kiikun lähtö voidaan kytkeä suoraan seuraavan siirtorekisterin tuloksi. Tämä käytännössä mahdollistaa sen, että siirtorekistereitä voidaan kytkeä sarjaan loputtomasti. Täytyy kuitenkin ottaa huomioon, että rekistereiden täyttyminen kestää sitä kauemmin, mitä enemmän niitä on sarjassa. Kuviossa 4 SIPO-tyyppisen siirtorekisterin sisäinen kytkentä, jossa sarjamuotoinen data syötetään SI-nastaan ja sille tehdään sarja/rinnan muunnos, jonka jälkeen rinnakkaisportit Qa – Qd asettuvat syötetyn ohjauksen mukaisiin tiloihin. Siirtorekistereissä on myös yleensä CLR-nasta, jota käytetään rekisterin tyhjentämiseen (kuvio 4).



KUVIO 4. SIPO-tyyppisen siirtorekisterin sisäinen kytkentä. (Learning electronics)

5.4 I²C LCD-näyttö

On olemassa paljon erilaisia mikrokontrollerialustoja varten tarkoitettuja näyttömoduuleita. Yksi yleisesti paljon käytetty näyttömoduuli on 16 sarakkeinen ja 2 rivinen LCD-moduuli (kuva 3). Tämä moduuli on suosittu sen halvan hinnan ja toimintavarmuuden takia. Lisäksi sitä on helppoa käyttää yhdessä Arduinon kanssa, sillä sille löytyy erittäin monipuolinen ja helppokäyttöinen ohjelmakirjasto. Tähän näyttömoduuliin on saatavissa myös I²C-LCD-sovitinlevy, joka näkyy kuvassa 3 näytön takana. Tällä I²C-LCD-sovitinlevyllä näyttöä voidaan käyttää I²C-protokollaa hyödyntäen. Myös tähän käyttötarkoitukseen Arduinolle löytyy erittäin kattava ohjelmakirjasto.

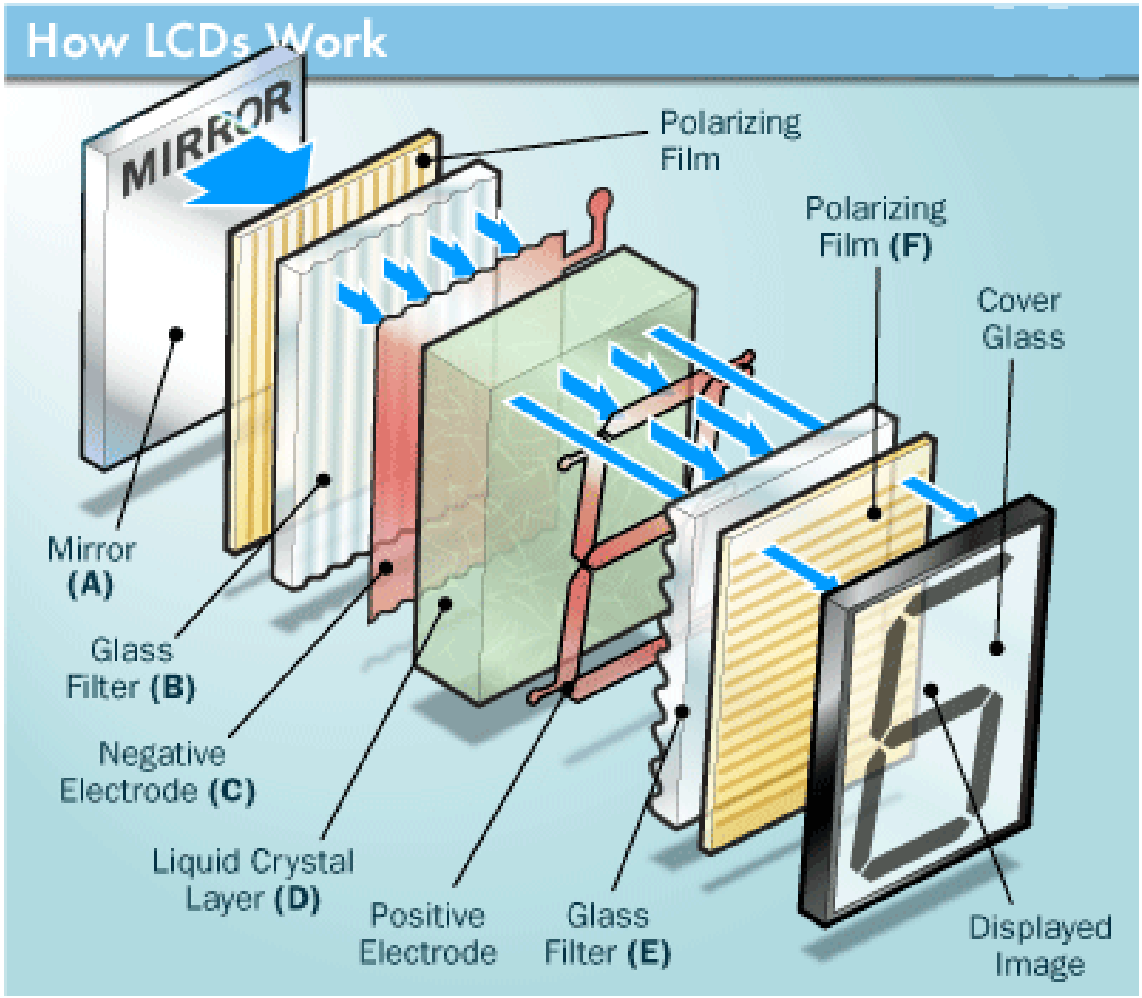


KUVA 3. I²C-protokollaa käyttävä 2x16 merkkinen LCD-näyttö (Partco)

5.4.1 LCD-näyttö

LCD eli Nestekidenäyttö on paljon käytetty näyttötyyppi, jonka toiminta perustuu valon polaroimiseen sähkökentällä. Nestekidenäyttö nimensä mukaisesti koostuu pienistä nesteeseen upotetuista kristallikiteistä. Näiden kiteiden asentoa voidaan muuttaa niihin kohdistuvaa sähkökenttää muuttamalla. Tämä kiteen asennon muuttuminen puolestaan muuttaa läpi pääsevän valon kulmaa, josta seuraa pikseleiden näkyminen näytöllä.

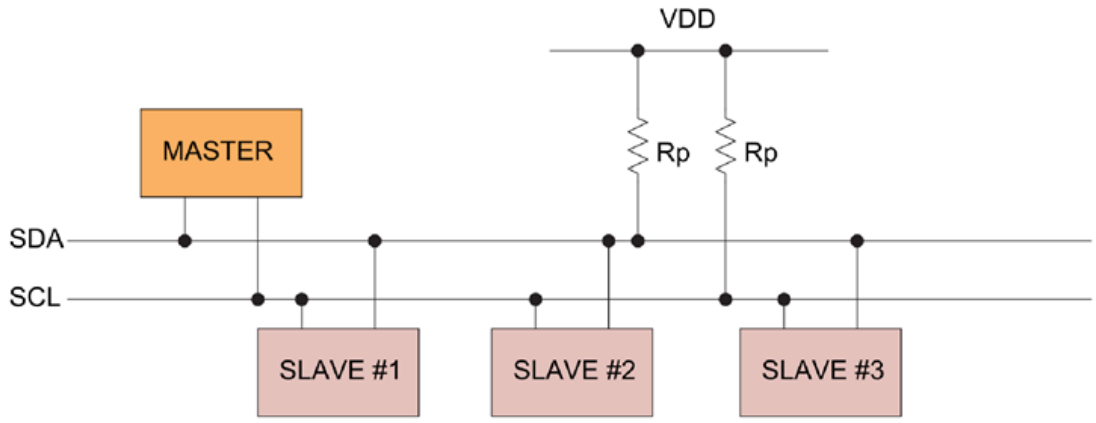
Kun kaksi kalvoa polarisoidaan ja asetetaan 90 asteen kulmaan toisiinsa nähden, valo ei pääse kulkemaan kalvojen läpi. Kun taas kalvojen väliin asetetaan nesteessä vapaasti olevia kiertyneitä nemaattisia kristalleja, ainakin osa valosta kiertyy 90 astetta kristalleja läpäistessään ja näin ollen läpäisee molemmat polarisoidut kalvot. Jos nesteeseen luodaan sähkökenttä, tuomalla sen molemmille puolille jännitelähteeseen kytketyt elektrodit, nesteessä olevat kristallit asettuvat sähkökentän suuntaisesti, eivätkä enää käännä valoa. Tästä seuraa, että nestekidenäytössä näkyy elektrodin muotoinen kuvio mustana, sillä valo ei läpäise tätä kohtaa näytöstä. Levyjen taakse voidaan asettaa peili, jolloin taustavaloa ei tarvita, vaan ulkopuolelta tuleva valo yksinkertaisesti heijastetaan takaisin. (kuvio 5.)



KUVIO 5. LCD-näytön rakenne (CircuitsToday)

5.4.2 I²C-protokolla

I²C-protokolla on Philipsin kehittämä sarjaliikenne-protokolla, jossa tietoa siirretään ainoastaan kahta ylösvetovastuksella varustettua linjaa SDA (Serial Data) ja SCL (Serial Clock) pitkin. Molemmat linjat ovat kaksisuuntaisia eli niissä voidaan siirtää dataa molempiin suuntiin. Protokolla on yksinkertainen, mutta hieman hitaampi tapa siirtää sarjamuotoista dataa usean eri slave- ja master-laitteen välillä, jos sitä verrataan vaikkapa SPI-protokollaan. I²C-protokolla onkin täydellinen valinta, kun halutaan esimerkiksi yhdistää mikrokontrolleriin useita erilaisia antureita tai näyttöjä, sillä se ei varaa kuin kaksi GPIO-nastaa ja sen toimintaperiaate on hyvin yksinkertainen. Tyypillisesti väylään kytkettyjen ylösvetovastusten arvo 10 kohm, kun väylää käytetään vakiotaajuudella 100 kHz. (kuvio 6.)

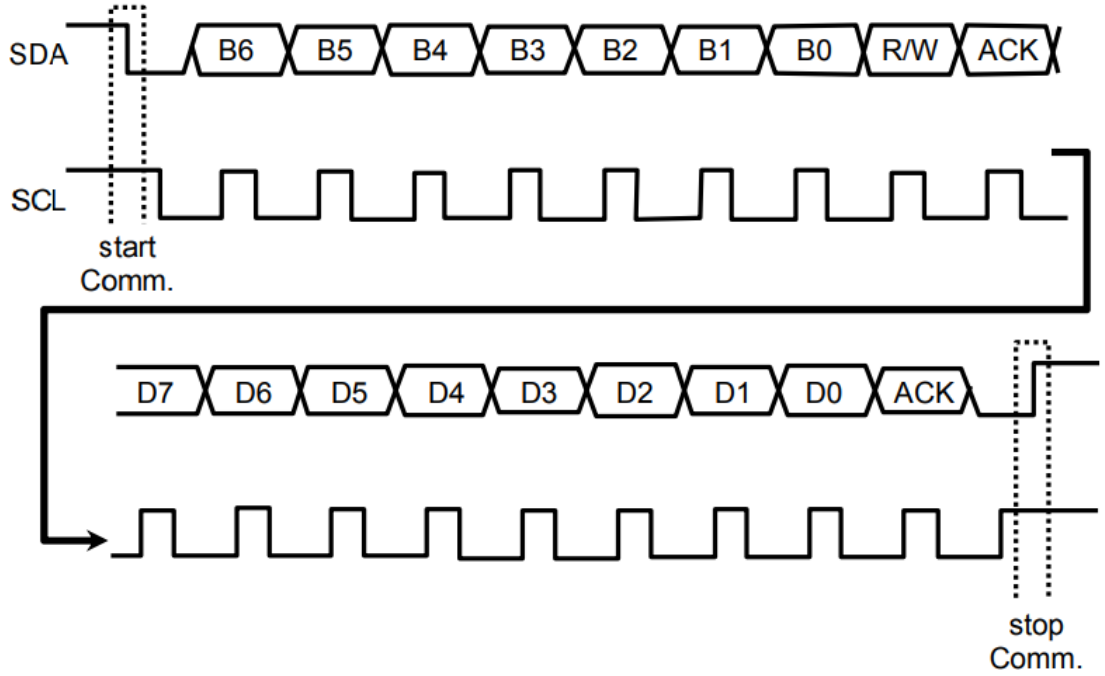


KUVIO 6. I²C-väylän periaatekuva (Afzal, S. 2020)

Väylän hitaus johtuu siitä, että kaikki liikenne tapahtuu vain kahta linjaa pitkin, joten tiedonsiirtoa voidaan suorittaa samaan aikaan vain kahden laitteen välillä. Jotta tiedonsiirto voidaan edes aloittaa, on master-laitteen lähetettävä sen laitteen osoite, johon halutaan yhteys muodostaa. Lisäksi on lähetettävä tieto siitä, että halutaanko slave-laitetta lukea vai kirjoittaa. Vasta kun laitteet ovat kätelleet, tiedonsiirto voi alkaa.

Väylään lähettävä laite aloittaa lähetyksen alkuehdolla, jossa SCL-nastaa pidetään ylhäällä SDA-linjan laskureunan aikana. Tätä seuraa halutun laitteen osoitteen lähetys. Laiteosoite on 7 bittiä pitkä ja sitä seuraa yksi bitti, joka kertoo, halutaanko laitetta lukea vai kirjoittaa. Tästä seuraa se, että kaikki linjaan kytketyt

laitteet vertaavat omaa laiteosoitettaan master-laitteen lähettämään osoitteeseen, jos osoitteet eivät täsmää laitteet siirtyvät odottamaan loppuehtoa. Jos taas osoite täsmää, niin slave-laite kuittaa olevansa valmiina lähettämään tai vastaanottamaan. Kuittauksen jälkeen aloitetaan tiedonsiirto, kunnes master-laite suorittaa loppuehdon, jossa SCL-linja pidetään ylhäällä SDA-linjan nousureunan aikana. Loppuehdon jälkeen muut linjaan kytketyt laitteet heräävät aktiivisesti odottamaan uutta yhteydenottoa. (Postolache, O. Silva Girão, P & Dias Pereira, J.M. 2010.) (kuvio 7.)



KUVIO 7. I²C-protokollan ohjaussignaalit (Postolache, O. Silva Girão, P & Dias Pereira, J.M. 2010)

6 SUUNNITTELU

6.1 Toimintaperiaate

Koulutettu asentaja kykenee mittaamaan ja toteamaan kaapelin kunnon käyttäen yleismittaria. Testausjärjestelmän toimintaperiaatetta lähdettiin suunnittelemaan siltä pohjalta, että miten asentajan tekemä mittausprosessi voitaisiin automatisoida, ja miten hänen työpanoksensa voitaisiin mahdollisimman hyvin korvata käyttämällä mikrokontrolleria. Tämä poistaisi inhimillisen virheen mahdollisuuden ja, mikrokontrollerin laskentatehon vuoksi, testaamiseen kuluva aika pienenisi murto-osaan siitä, mitä se on tällä hetkellä. Tällä hetkellä kaapelin toimintakunnon toteaminen tapahtuu mittaamalla yleismittarilla jokaisen kaapelissa olevan liittimen jokaisen nastan yhteys kaikkiin muihin nastoihin kaikissa kaapelin liittimissä. Mittauksen aikana, tai sen jälkeen, mitattuja arvoja verrataan ehjän kaapelin arvoihin ja sitä kautta todetaan kaapelin kunto.

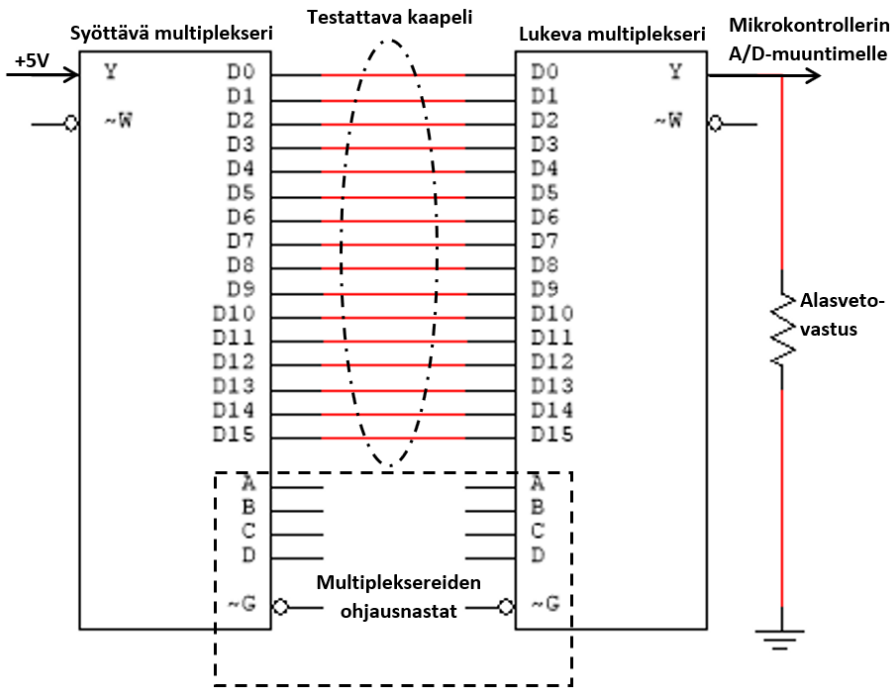
Toimintaperiaatteen suunnittelu aloitettiin ideasta kytkeä testattava kaapeli suoraan mikrokontrollerin digitaalisten ja analogisten porttien väliin niin, että kaikki kaapelin toisen pään liittimien nastat olisivat syöttäviä nastoja eli digitaalisia ja toisen pään taas lukevia eli analogisia. Sitten digitaaliset nastat kirjoitettaisiin yksitellen 5 volttiin ja samalla asetettaisiin loput digitaaliset nastat korkeaimpedanssiseen tilaan kuten digitaaliseksi sisäänmenoksi. Tämän jälkeen mikrokontrollerilla mitattaisiin jokainen kaapelin toisessa päässä oleva analoginen nasta ja tallennettaisiin mittatulokset muuttujaan. Muuttuja, johon mitatut tulokset tallennettaisiin voisi olla kaksiulotteinen taulukko eli matriisi. Matriisin rivi, jolle mitattu tulos tallennettaisiin, vastaisi aina sen digitaalisen nastan numeroa, johon 5 volttia kirjoitettiin ja jokainen sarake vastaisi luettua analogista nastaa, josta sen hetkinen mittatulos luettiin. Kun kaikki analogiset nastat olisi luettu ja tulokset tallennettu, siirryttäisiin kirjoittamaan seuraava digitaalinen nasta ylös. Tätä prosessia toistettaisiin, kunnes kaikki digitaaliset nastat olisi käyty läpi. Lopputuloksena saataisiin matriisi, joka pitäisi sisällään kaikki testattavan kaapelin liitinten väliset kytkennät.

Yllä mainitun idean pohjalta alettiin pala kerrallaan jalostamaan toimintaperiaatetta, joka täyttäisi kaikki testausjärjestelmälle asetetut vaatimukset. Toimintaperiaatetta alettiin rakentaa Arduino Nano V3 kehitysalustan ympärille, koska se on erittäin monipuolinen alusta ja Arduino IDE-ohjelma oli osoittautunut hyväksi kehitysympäristöksi jo aikaisemmissa projekteissa. Ensimmäinen suurempi ongelma oli se, että Arduino Nano V3 alustassa on vain 8 analogista nastaa, eivätkä digitaalisetkaan nastat tulisi riittämään 33-johtimisen kaapelin kytkemiseen. Tästä syystä tarvittiin keino, jolla kyseisen alustan nastat saataisiin riittämään.

6.1.1 Multipleksereiden hyödyntäminen

Mikrokontrollerin nastojen riittämättömyyden ratkaisemiseksi heräsi ajatus käyttää multipleksereitä nastojen jakamiseen, jolloin saataisiin käytettyä samaa mikrokontrollerin nastaa ohjaamaan useampaa linjaa. Ajatuksena oli, että multipleksereitä voitaisiin käyttää kytkinkomponentteina, joita mikrokontrolleri ohjaisi digitaalisilla signaaleilla. Näin pystyttäisiin vähentämään mikrokontrollerilta vaadittavien nastojen määrää. Kuvio 8 havainnollistaa kahden 16-linjaisen multiplekserin käyttöä testisignaalin syöttämiseen ja lukemiseen testattavasta kaapelista. Tällä kytkentäperiaatteella 16 digitaalisen ja 16 analogisen nastan sijaan, selvitään 10 digitaalisella ja yhdellä analogisella nastalla aiemman toiminnallisuuden pysyessä samana.

Kuvion 8 kytkentäperiaatetta ja multipleksereitä käyttämällä mikrokontrolleri voidaan ohjelmoida ohjaamaan mikä tahansa testattavan kaapelin johdin 5 volttiin ja myös lukemaan kaikki kaapelin linjat käyttäen vain yhtä analogista nastaa. Kuviosta 8 nähdään, että lukevan multiplekserin lähtöön on kytkettynä alasvetovastus, jossa tapahtuvaa testisignaalin aiheuttamaa jännitehäviötä mitataan mikrokontrollerilla. Kun tällaisia multiplekseripaketteja kytketään 3 rinnakkain, mahdollistetaan 48-johtimisen kaapelin testaaminen, mikä riittää täysin tämän työn tarpeisiin. Multipleksereiden ohjausnastojen ohjaamiseen vaadittaisiin silti useampi digitaalinen nasta, ja esimerkiksi Arduino Nanossa digitaalisten nastojen määrä on hyvinkin rajallinen. Tästä syystä pyrittiin löytämään vieläkin parempi keino pienentää mikrokontrollerilta vaadittavien digitaalisten nastojen määrää.



KUVIO 8. Multipleksereiden kytkennän hahmotelma

6.1.2 Siirtorekistereiden hyödyntäminen

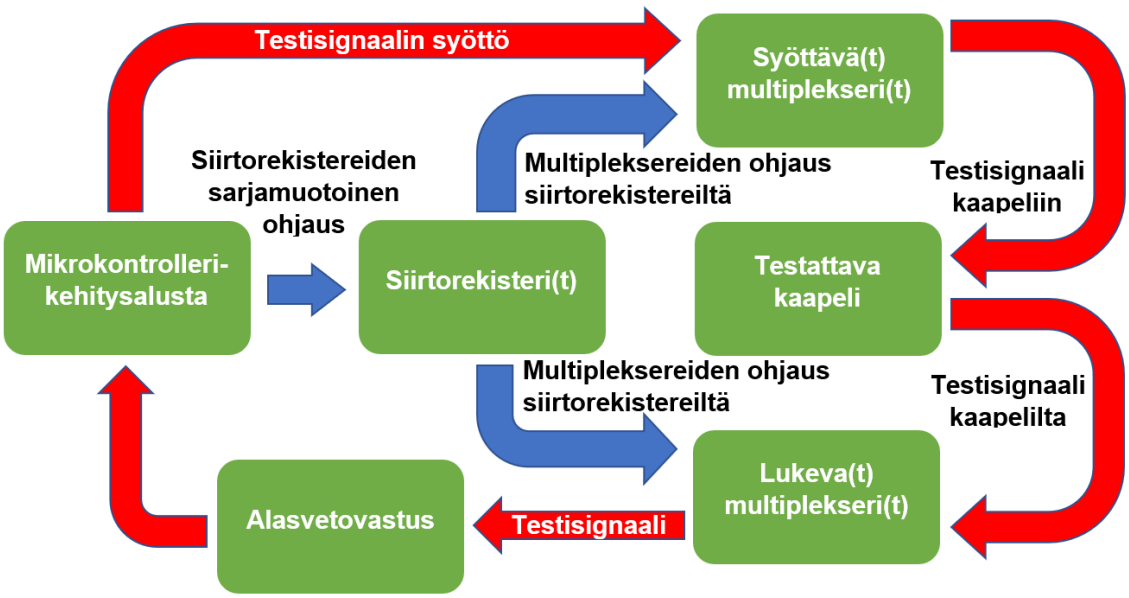
Tarvittiin keino ohjata digitaalisesti useampaa multiplekseriä käyttäen mahdollisimman pientä määrää mikrokontrollerin digitaalisista nastoista. Tästä heräsi idea, että mikrokontrollerilta saatava ohjaus voisi olla sarjamuotoinen, jonka jälkeen ohjaukselle voitaisiin tehdä sarja/rinnan -muunnos käyttäen ulkoisia komponentteja. Näin mikrokontrollerilta vaadittavien nastojen määrä saataisiin pysymään mahdollisimman alhaisena. Siirtorekistereitä voidaan käyttää sarja/rinnan muunnoksen tekemiseen ja niiden ohjaaminen mikrokontrollerilla on suhteellisen yksinkertaista ja helppoa. Lisäksi siirtorekistereitä on helppoa ketjuttaa. Tämä mahdollistasi sen, että vain muutamalla mikrokontrollerin digitaalisella nastalla voitaisiin ohjata useita kymmeniä tai vaikka satoja digitaalisia lähtöjä.

Järjestelmän toimintaa suunniteltiin niin, että mikrokontrolleri antaisi sarjamuotoisen ohjauksen siirtorekistereille, jotka taas jakaisivat ohjauksen eteenpäin multipleksereille. Multipleksereiden ohjaaminen siirtorekistereiden kautta tuottaa kuitenkin viivettä ja näin ollen hidastaa testaamista. Tämä johtuu siitä, että mikrokontrollerilta saadun sarjamuotoisen signaalin tulee tässä käyttötarkoituksessa

siirtää myös aikaisempi ohjaus pois rekistereiden lähdöstä. Toisin sanoen siirtorekisterit täytyy aina kirjoittaa täyteen tai muuten vanhan ohjauksen bitit vain siirtyvät eteenpäin annetun ohjauksen verran. Toisaalta kaikkien multipleksereiden ohjaamiseen tarvitaan vain noin 16 bittiä, eli vaikka ohjaus tarvitsee kirjoittaa joka kerta kokonaisuudessaan, niin käytännössä viiveet ovat sen verran pieniä, että ne eivät vaikuta merkittävästi tämän työn vaatimuksiin.

6.1.3 Kehitetty toimintaperiaate

Toimintaperiaate alkoi muodostua hiljalleen selkeäksi kokonaisuudeksi. Kuvio 9 havainnollistaa suunniteltua toimintaperiaatetta. Punaiset nuolet osoittavat testisignaalin reittiä kytkennän läpi ja siniset nuolet esittävät ohjaussignaaleita. Kuvion 9 vihreät laatikot edustavat fyysisiä komponentteja. Mikrokontrolleri antaa sarjamuotoisen ohjauksen siirtorekistereille. Siirtorekisterit jakavat ohjaukset edelleen syöttävien ja lukevien multipleksereiden ohjausnastoille, minkä seurauksena multipleksereiden kytkimet ohjaavat testisignaalin testattavasta kaapelista valitun johtimen lävitse. Tämän jälkeen mikrokontrolleri mittaa testisignaalin alasvetovastuksessa aiheuttaman jännitehäviön ja tätä prosessia toistetaan, kunnes kaikki kaapelin johtimet on mitattu. Tällä toimintaperiaatteella aika, joka kului siirtorekistereiden ohjauksesta multipleksereiden sisäisten kytkimien asettumiseen oli vain muutamia mikrosekunteja. (kuvio 9.)



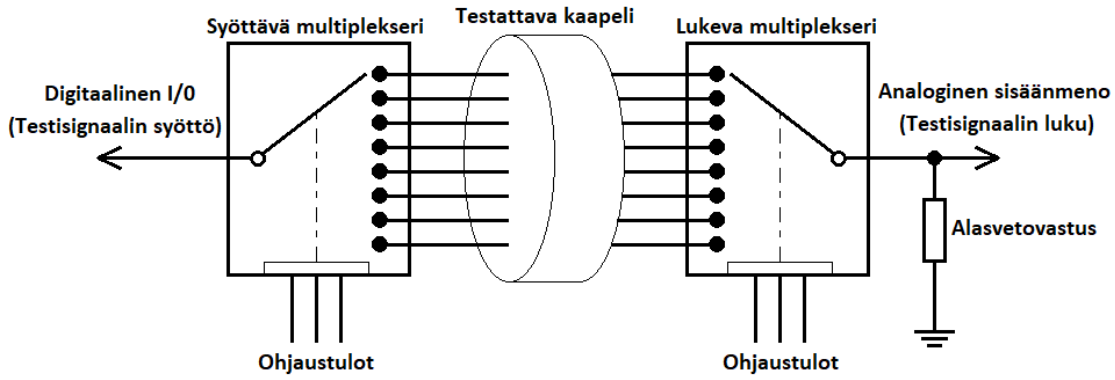
KUVIO 9. Laitteen toimintaperiaatetta hahmottava kaavio

6.2 Versiot

Laitteesta päädyttiin suunnittelemaan kaksi versiota, sillä ensimmäisen version toiminnallisuus todettiin sen prototyyppiä testattaessa riittämättömäksi, ja koska sillä ei pystytty toteamaan kaapelin kuntoa riittävän tarkasti. Versioiden välillä tapahtui suuri muutos; ensimmäisen version älynä toiminut Arduino Nano V3 ei yksinkertaisesti riittänyt teholtaan taikka kapasiteetiltään suorittamaan testiä tarpeeksi nopeasti ja varastoimaan tarpeeksi mittatuloksia. Tämän takia siirryttiin käyttämään tehokkaampaa 32- bittistä STM32-alustaa. Lisäksi uusi versio mahdollisti kaapelin molempien päiden sisäisten yhteyksien lukemisen, johon ensimmäinen versio ei pystynyt. Tämä muutos kuitenkin tuplasi vaadittavien multiplekserein määrän ja mikrokontrollerilta vaadittiin huomattavasti enemmän laskentatehoa ja muistia. Uuteen versioon lisättiin myös useita erilaisia lisäominaisuuksia, joita ensimmäisestä versiosta ei löytynyt.

6.2.1 Ensimmäisen versio

Ensimmäisessä versiossa ideana oli syöttää testisignaali multiplekserin kautta kaapelin toisesta päästä linja kerrallaan sisään testattavaan kaapeliin. Tämän jälkeen kaapelin toisesta päästä mitattiin jokainen linja, käyttäen toista multiplekseriä. Tuloksista pystyttiin ohjelmallisesti päättelemään, mitkä kaapelin linjat olivat johtaneet testisignaalin lävitsensä (kuvio 10). Sitten tulosta verrattiin ohjelmallisesti ehjän kaapelin kytkentään ja todettiin kaapelin toimintakunto. Ensimmäiseen versioon tuli yhteensä 6 kappaletta 16 kanavaisia multipleksereitä ja kaksi 8-bittistä siirtorekisteriä, joita käytettiin jakamaan mikrokontrollerilta tuleva sarjamuotoinen ohjaus multipleksereiden ohjausnastoille. Tällä versiolla pystyttiin siis testaamaan 48- linjaisia kaapeleita.



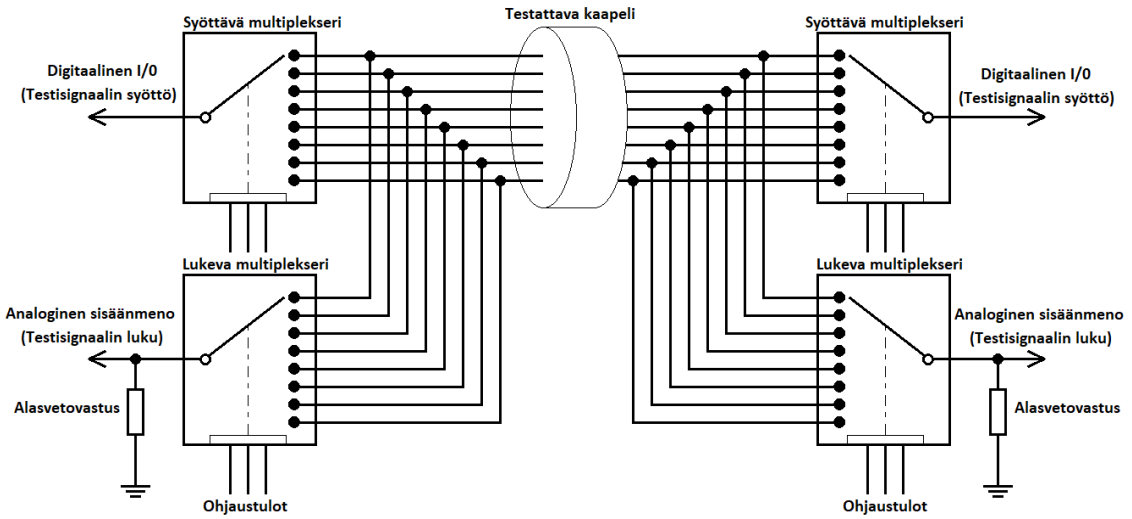
KUVIO 10. Ensimmäisen version mittauskytkennän periaatekuva

Tätä versiota ei kuitenkaan lähdetty kehittämään valmiiksi tuotteeksi asti sen ominaisuuksien riittämättömyyden takia ja se toimi lähinnä prototyyppinä toiselle uudemmalle versiolle. Lisäksi tällä versiolla saatiin mitattua vain testattavan kaapelin liitinten väliset yhteydet, eikä kaapelin molempien päiden sisäisiä kytkentöjä. Tässä kohtaa tultiin siihen tulokseen, että myös liittimien sisäiset kytkennät tulisi pystyä mittaamaan, sillä jos liittimessä on esimerkiksi sisäinen oikosulku kahden tai useamman nastan kanssa, mutta ei yhteyttä toisen pään liittimeen, niin testisignaali ei pääse kulkeutumaan kaapelin lävitse, eikä testeri näin ollen pysty havaitsemaan sisäistä yhteyttä. Tällä versiolla saatiin kuitenkin testattua multiplekserein, siirtorekistereiden ja ohjauselektroniikan toimintaa sekä yleisesti kytkennän toimintaperiaatetta. Lisäksi suurin osa ohjelmasta kirjoitettiin aluksi tähän versioon ja sitten myöhemmin siirrettiin uudempaan versioon.

6.2.2 Toinen ja lopullinen versio

Testattavan kaapelin molempien päiden sisäisten kytkentöjen määrittämiseksi jouduttiin tekemään suuriakin muutoksia alun toimintaperiaatteeseen, mutta pääperiaate saatiin silti pysymään melko samanlaisena. Jotta myös liitinten sisäiset kytkennät saataisiin mitattua, täytyi testisignaalin syöttö pystyä toteuttamaan niin, että olisi mahdollista ohjelmallisesti päättää kummasta päästä kaapelia signaali syötetään sisään. Lisäksi täytyi saada mitattua mikrokontrollerilla jännite kaikista testattavassa kaapelissa olevista linjoista. Tämä ratkaistiin tuplaamalla multiplekserein määrä niin, että testattavan kaapelin molempiin päihin kytkettiin syöttävä sekä lukeva multiplekseri. Tämä mahdollisti sen, että testisignaali kyettiin

kytkemään testikaapeliin sen molemmista päistä ja kaapelin molempien päiden kaikki linjat pystyttiin lukemaan samalla kertaa. (kuvio 11.)



KUVIO 11. Toisen ja lopullisen version mittauskytkennän periaatekuva

Tässä versiossa mikrokontrollerilta vaadittava muuttujille tarkoitetun muistin määrä kuitenkin kolminkertaistui. Tämän sekä Arduinon hitauden takia päädyttiin vaihtamaan mikrokontrollerialustaa. Arduinon Nanon sijaan päädyttiin käyttämään STM32-mikrokontrollerialustaa, joka on huomattavasti nopeampi ja siinä on kymmenkertainen määrä muistia Nanoon verrattuna. Tähän versioon päädyttiin lisäämään myös SD-kortti, asetusten ja kaapeleiden kytkentöjen tallentamista varten. STM32-mikrokontrollerissa on sisäänrakennettu kellorekisteri, joten myös tämä ominaisuus hyödynnettiin lisäämällä kellonäkymä.

Lisäksi uuden kehitysalustan valintaan vaikutti STM32-mikrokontrollerin A/D-muuntimen tarkkuus, sillä Arduinon 10-bittisen muuntimen sijaan saatiin käyttöön 12-bittinen muunnin, joka mahdollisti noin neljä kertaa tarkemman jännitemittauksen, jolla testin tarkkuus saatiin huomattavasti paremmaksi. Tämä johtuu siitä, että Arduinon 10-bittisellä A/D-muuntimella saadaan teoreettisesti mitattua vain noin 5 millivoltin jännitteitä, kun taas Bluepill-mikrokontrollerin 12-bittisellä muuntimella pystytään mittaamaan jopa 800 mikrovoltin jännitteitä. Tämä A/D-muuntimen teoreettinen tarkkuus saadaan laskettua, kun mikrokontrollerin jännitetaso jaetaan A/D-muuntimen askelmäärällä. Esimerkiksi Bluepill mikrokontrollerin jännitetaso on 3.3 volttia ja 12-bittisessä A/D-muuntimessa on 4096 askelta, jolloin sen tarkkuus on 3.3 volttia jaettuna 4096 askeleella eli noin 800 mikrovolttia.

6.3 Käyttöliittymä

Koska järjestelmän käytöstä haluttiin tehdä mahdollisimman helppoa ja yksinkertaista, täytyi käyttöliittymän suunnitteluun panostaa. Tarvittiin keino ohjata järjestelmää sekä tapa antaa visuaalista tietoa laitteen tilasta ja toiminnasta. Aikaisemmista projekteista tiedettiin, että I²C-väylässä toimiva 2x16 merkkinen LCD-näyttö on helppokäyttöinen ja Arduinon kanssa yhteensopiva näyttölaite. Tiedettiin myös, että tämän LCD-näytön käyttöä varten Arduinolle löytyy hyvä ja toimiva ohjelmakirjasto. Tästä syystä LCD-näyttö valittiin käytettäväksi näyttölaitteeksi. Laitteen ohjaamista varten valittiin käytettäväksi tavallinen painonappi. Käytön yksinkertaistamiseksi päädyttiin käyttämään vain yhtä ohjauspainiketta, ja jotta tällä yhdellä painonapilla saatiin tarpeeksi toiminnallisuutta, käsiteltiin napin painalluksen kestoa ohjelmallisesti niin, että aika jonka painike pidettiin painettuna, määritti ohjelmassa suoritettavan toiminnon.

Kun käyttöliittymän fyysiset komponentit oli saatu valittua, aloitettiin suunnittelemaan ohjelmallista käyttöliittymää. Helpoin ja selkein tapa toteuttaa tämä oli tehdä itseään kiertävä valikko, jossa jokainen otsikko vastaa tiettyä toimintoa ohjelmassa ja otsikon ollessa valittuna se tulostetaan LCD-näytölle. Valikon otsikoiden välillä liikutaan lyhyillä ohjausnapin painalluksilla ja aktiivinen eli sillä hetkellä näytöllä näkyvä otsikko voidaan valita pidemmällä painalluksella, jolloin suoritetaan kyseisen otsikon alla oleva aliohjelma.

6.4 Piirilevy

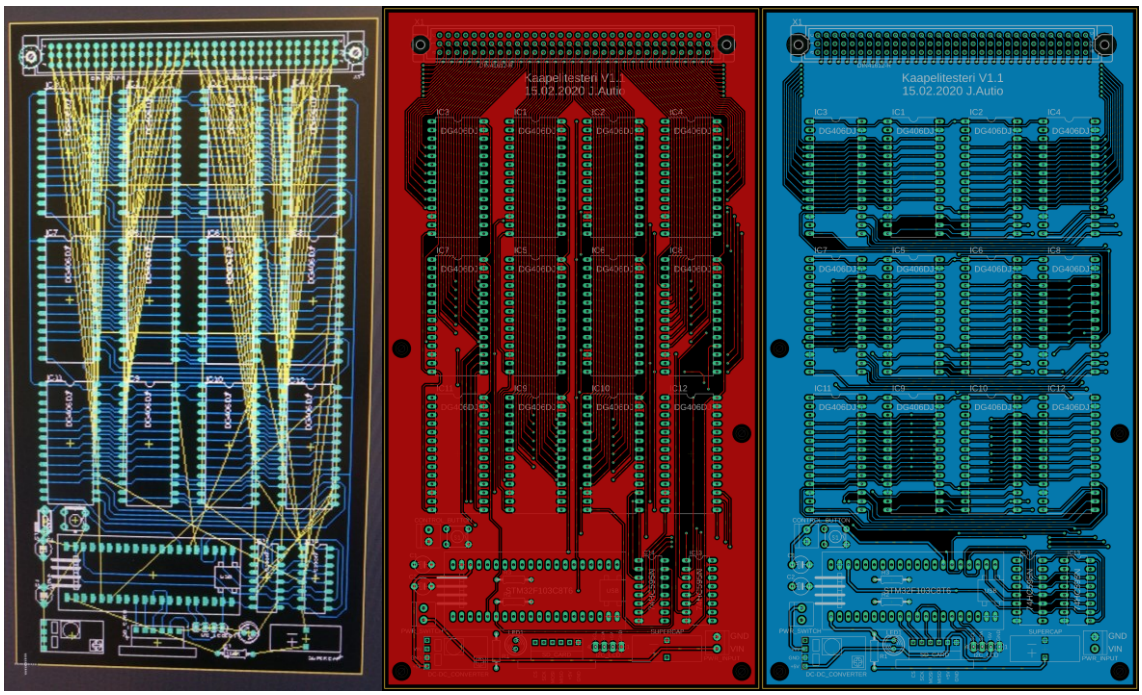
Piirilevy suunniteltiin Autodesk Eagle-ohjelmistolla. Piirilevyn suunnittelu jakautuu kahteen osioon, aluksi suunnitellaan kytkentäkaavio ja tämän jälkeen kytkentäkaavion perusteella suunnitellaan itse piirilevy. Eagle-ohjelmistolle löytyy laajat komponenttikirjastot ja ohjelmaan on tarvittaessa melko helppoa luoda uusia komponentteja.

Käsin hahmotellun kytkentäkaavion perusteella, piirrettiin kytkentä Eagle-ohjelmistoon. Kaikki muut komponentit löytyivät komponenttikirjastoista paitsi hakku-

rikortti, SD-kortin adapteri ja superkondensaattori. Nämä komponentit täytyi mallintaa ohjelmaan ja tallentaa käyttäjän omaan komponenttikirjastoon. Komponenttien lisääminen oli kuitenkin hyvin dokumentoitu, eikä sen opettelemiseen kulunut kauaa aikaa. Tämän jälkeen siirryttiin suunnittelemaan itse piirilevyn johdotusta. Komponentteja oli sen verran paljon, että levystä täytyi suunnitella kaksipuoleinen eli kupariset johdotukset suunniteltiin piirilevyn molemmille puolille. Piirilevylle täytyi myös suunnitella reiät sen kiinnittämistä varten.

Suunnitteluohjelma näyttää keltaisella viivalla, mitkä komponenttien nastat kuuluvat yhteen ja tämän pohjalta komponentit täytyy sijoittaa siten, että niiden johdottaminen on mahdollista. Tämä oli todella haastavaa, sillä tähän kytkennässä on kaikkiaan 12 multiplekseriä ja jokaisessa multiplekserissä on 28 nastaa. Lisäksi 96-nastainen liitin lattakaapelille oli haastavaa johdottaa siten, että sen nastat saatiin menemään loogisessa järjestyksessä. Lopulta kuitenkin päädyttiin komponenttisijoittelun kannalta hyvään ratkaisuun ja kaikki komponentit saatiin sopimaan suhteellisen pienellekin alueelle. Johdotus suunniteltiin ensiksi piirilevyn pohjaan kuten kuviosta 12 nähdään ja tämän jälkeen siirryttiin piirtämään loput johdotukset piirilevyn yläpuolelle.

Kun piirilevyn johdotukset oli saatu suunniteltua, haluttiin toteuttaa kuparitäyttö piirilevyn molemmille puolille. Kun kuparitäyttöön yhdistettiin piirilevyn maa, saatiin pienennettyä piirilevyn maapotentiaalin resistanssia eli vahvistettua sen maata ja parannettua levyn häiriön sietokykyä. Valmiit piirilevysuunnitelmat näkyvät kuviossa 12, jossa punainen väri on piirilevyn yläpuolelle piirrettyjä johdotuksia ja sininen piirilevyn pohjaan piirrettyjä johdotuksia.



KUVIO 12. Keskeneräinen piirilevysuunnitelma vasemmalla ja valmiit piirilevysuunnitelmat Autodesk Eagle-ohjelmistossa

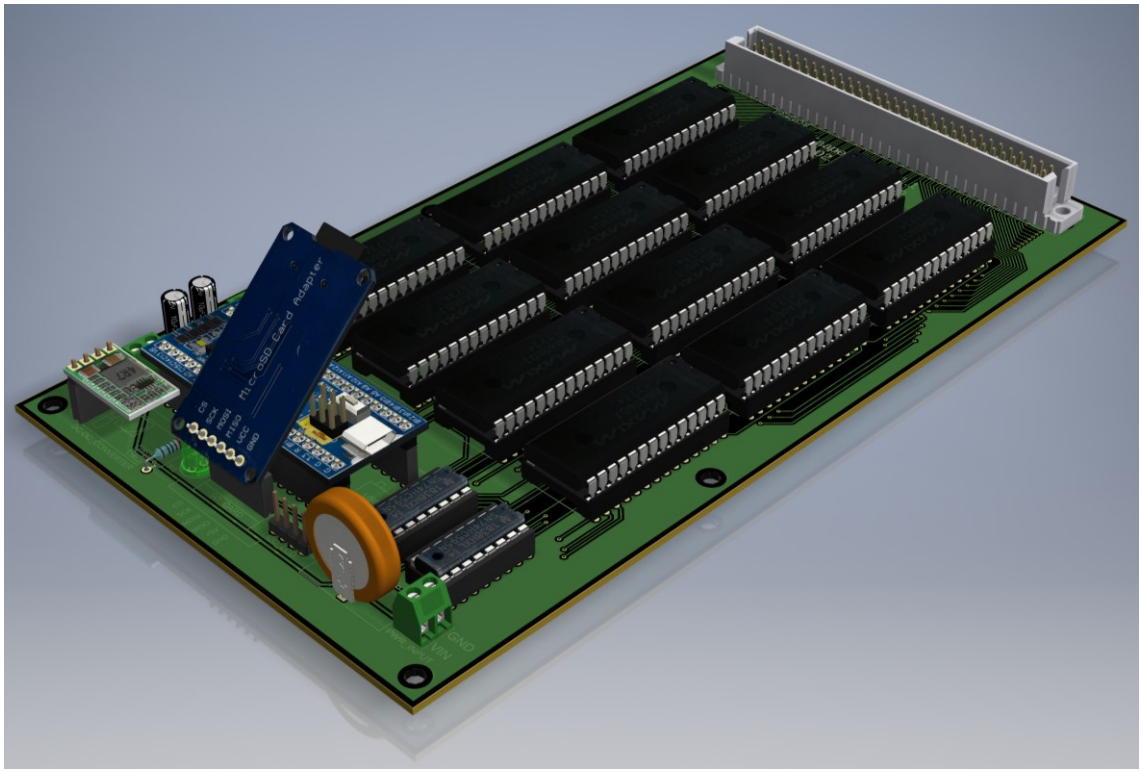
6.5 3D-mallinnus

Laitteen piirilevyt ja kotelot 3D-mallinnettiin Autodesk Inventor- suunnitteluohjelmistolla, joka on ammattikäyttöön tarkoitettu 3D-mallinnus ohjelmisto. Aluksi suunniteltiin piirilevyn 3D-malli hyödyntäen piirilevyn suunnitteluohjelmasta saatua levyn rei'ityksen ja koon näyttävää 3D-mallia. Loput komponentit 3D-mallinnettiin erikseen. Mallinnetuista komponenteista ja piirilevystä luotiin kokoonpanomalli, jonka perusteella alettiin suunnitella testausjärjestelmän keskusyksikölle 3D-tulostettavaa koteloa. Lopuksi vielä mallinnettiin järjestelmään sopiville adapterirasioille kokoonpanomallit 3D-tulostusta varten.

6.5.1 Piirilevyn kokoonpanomalli

Jotta myöhemmässä vaiheessa saataisiin suunniteltua laitteen keskusyksikölle kotelo, täytyi ensin 3D-mallintaa vähintäänkin tilamalli piirilevystä. Piirilevyn suunnitteluohjelma Autodesk Eagle -ohjelmisto mahdollisti piirilevyn ja sen rei'ityksen sisällään pitävän 3D-mallin viemisen Autodek Inventor-ohjelmistoon, mutta muut

komponentit oli mallinnettava erikseen. Komponentit päädyttiin mallintamaan ehkä hieman liiankin yksityiskohtaisesti työntömittaa ja komponenttien datalehtiä hyödyntäen. Komponentit mallinnettiin yksitellen ja lopuksi piirilevystä tehtiin kokoonpanomalli, jonka ympärille voitiin ruveta suunnittelemaan koteloa keskusyksikköä varten. Kaiken kaikkiaan piirilevyn kokoonpanomallista tuli todella realistinen (kuvio 13). Tämä helpotti huomattavasti kotelon suunnittelua, sillä mallista nähtiin suoraan piirilevyn vaatima tila. Tämä mahdollisti kotelon suunnittelun samalla, kun odotettiin tilatun piirilevyn saapumista.



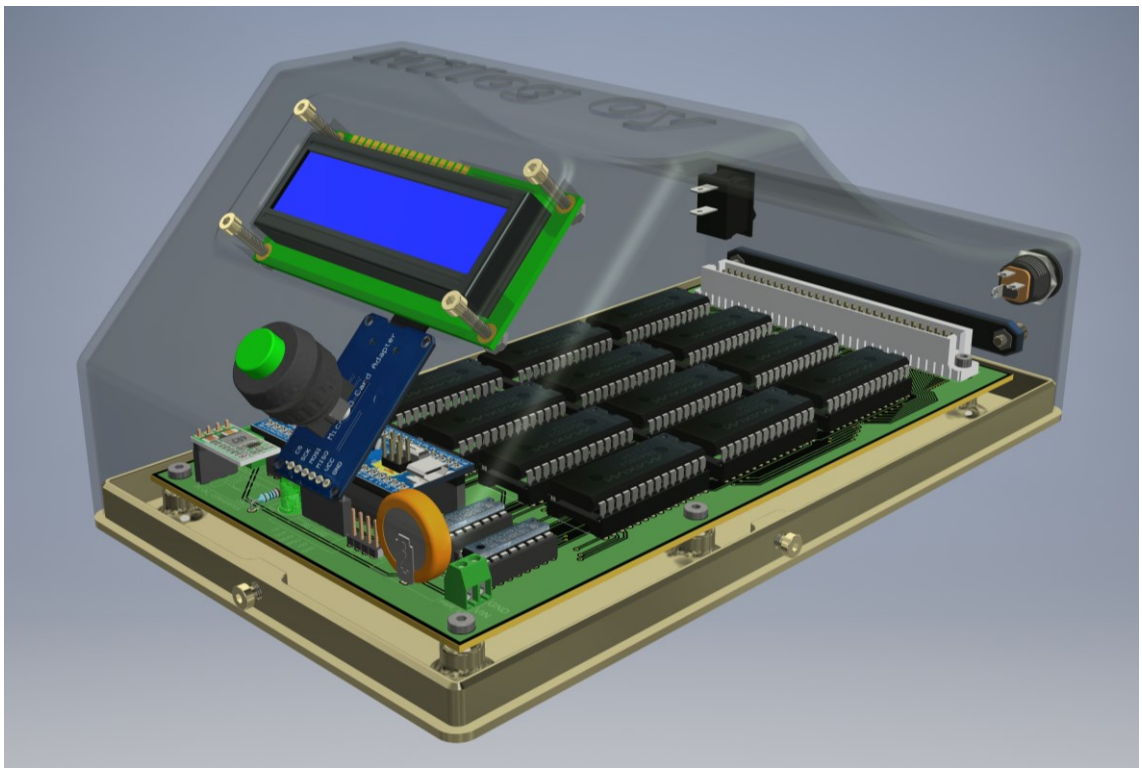
KUVIO 13. Piirilevyn kokoonpanon 3D-malli

6.5.2 Keskusyksikön kokoonpanomalli

Keskusyksikön kotelosta haluttiin tehdä yksinkertainen ja selkeä. Lisäksi mallinnettaessa täytyi ottaa huomioon se, että kotelon valmistamiseen haluttiin hyödyntää 3D-tulostusta. Tulostusteknisistä syistä haluttiin minimoida tarvittavan tukirakenteen määrä. Näistä syistä malli pyrittiin suunnittelemaan helpoilla ja selkeillä ratkaisuilla niin, että se olisi mahdollisimman helppo tulostettava. Kotelo päädyttiin valmistamaan kahdesta osasta: kannesta ja pohjasta.

Kotelon 3D-mallia alettiin suunnitella valmiin piirilevyn kokoonpanomallin ympärille. Mallinnus aloitettiin pohjalevystä. Pohjalevy mitoitettiin sopivasti piirilevyä suuremmaksi ja siihen mallinnettiin kiinnikkeet piirilevyä varten. Piirilevyn kiinnittämistä varten haluttiin käyttää tulosteeseen upotettavia kierreholkkeja, joten myös sellaiset oli mallinnettava ja tuotava kokoonpanomalliin. Kun pohjalevyyn oli saatu kiinnikkeet piirilevylle, piirilevyn kokoonpanomalli lukittiin kiinni pohjalevyyn, minkä jälkeen voitiin siirtyä suunnittelemaan kantta sen ympärille.

Kantta varten täytyi kuitenkin vielä mallintaa kaikki koteloon kiinnitettävät käyttölaitteet kuten painonappi, näyttö, virtakytkin ja virtaliitin. Myös ne mallinnettiin työntömitan ja komponenttien datalehtien avulla. Virtaliitin ja virtakytkin sijoitettiin kannen takaosaan sopivalle korkeudelle, jotta lattakaapelin liitin mahtuu pystysuunnassa piirilevyllä olevaan liittimeen. Näyttö ja ohjauspainike asetettiin 45 asteen kulmaan kotelon etuosaan, jotta näyttö olisi sopivassa kulmassa istuvaan tai seisovaan asentajaan nähden. Lopuksi suunniteltiin kannen ja pohjan kiinnitystapa. Tässä päädyttiin siihen ratkaisuun, että pohjan sisäseiniin tehtiin upotukset muttereille ja kanteen reiät samoihin kohtiin. Sitten kansi ja pohja ruuvattiin yhteen jokaiselta sivulta käyttäen koneruuveja. (kuvio 14)

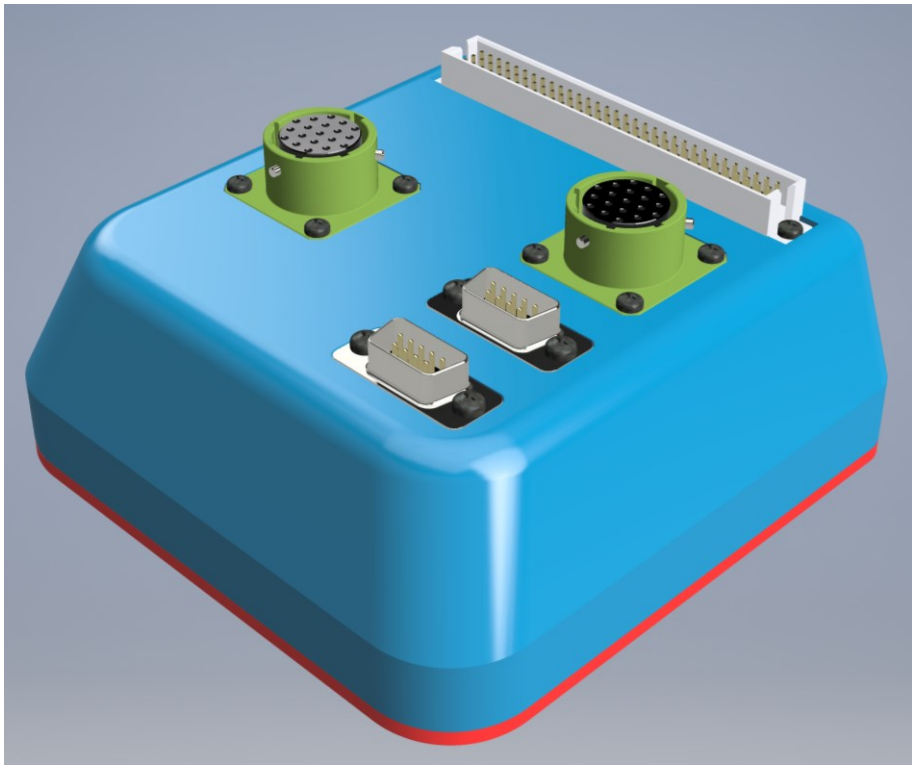


KUVIO 14. Keskusyksikön kokoonpanon 3D-malli läpinäkyvällä kannella

6.5.3 Adapterirasioiden kokoonpanomallit

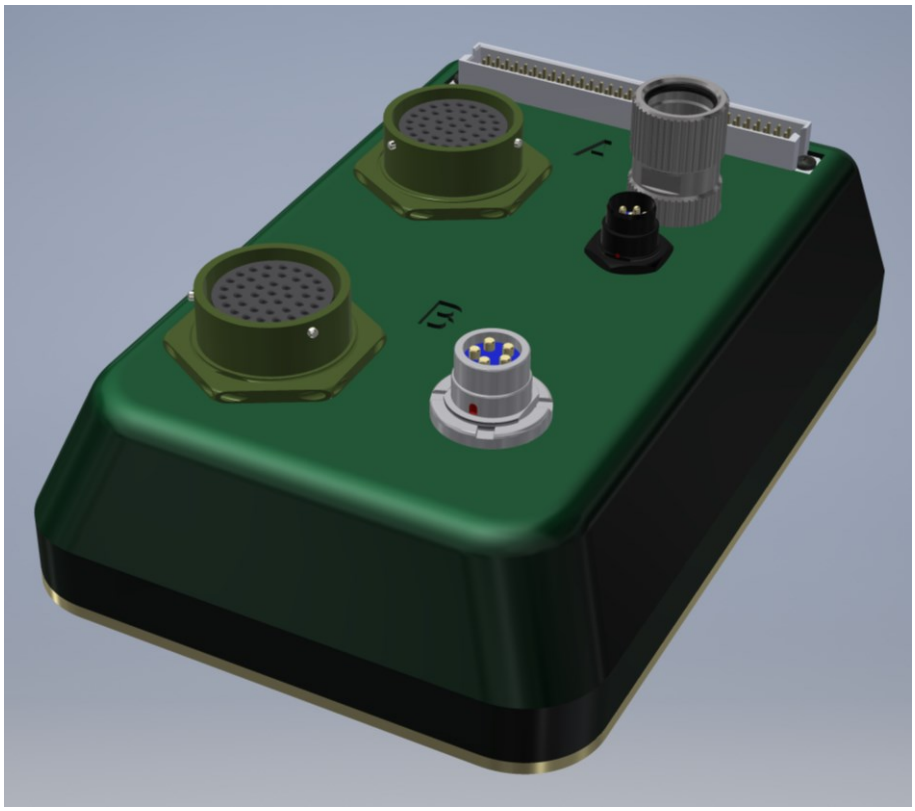
Jotta keskusyksikköön saataisiin liitettyä erityyppisiä ja eri liittimillä varustettuja kaapeleita, täytyi järjestelmään suunnitella adapterirasioita, jotka toimivat liitäntärajapintana testattavan kaapelin ja testausjärjestelmän keskusyksikön välissä. Näiden rasioiden 3D-mallintamisessa tuli ottaa huomioon se, että niiden tuli kestää melko kovaa rasitusta. Etenkin sotilaskäyttöön tarkoitetut liittimet ovat erittäin tiukkaan istuvia, joten adapterirasiaan kiinnitettyjen liittimien tuli myös olla tehokkaasti kiinni, jottei adapterirasia hajoa kaapelia kytkettäessä tai pois otettaessa. Myös rasioiden pohjan pinta-ala piti pyrkiä maksimoimaan, jotta kytketyn kaapelin massa ei kaataisi rasiaa. Haluttiin kuitenkin, että adapterirasiat olisivat suhteellisen kompaktin kokoisia säilytyksen ja kuljettamisen helpottamiseksi.

Aluksi mallinnettiin kaikki adapterirasiaan tarvittavat liittimet, jonka jälkeen mallinnettiin rasian ulkokuori. Kuoresta päätettiin tehdä 8 mm paksu, jotta se varmasti kestäisi kovaa rasitusta. Kun kotelon ulkokuori oli saatu mallinnettua, siihen tehtiin reiät ja upotukset halutuille liittimille. Lopuksi vielä mallinnettiin pohjalevy, johon tuli ruuvattavat kiinnikkeet. Lopputuloksena syntyi kuviossa 15 esitetty kokoonpanomalli.



KUVIO 15. Ensimmäisen adapterirasian 3D-malli

Ensimmäinen adapterirasian malli ei kuitenkaan ollut tarpeeksi iso, jotta siihen olisi voitu kytkeä vieläkin isompia liittimiä tiedonsiirtokaapeleita varten. Tästä syystä seuraavan adapterirasian pituutta kasvatettiin huomattavasti, mutta muuten malli pysyi lähes samanlaisena kuin aikaisemmassakin adapterirasiassa. Tähän malliin päätettiin vielä lisätä kirjaimet A ja B kuvaamaan sitä, mitkä liittimet on kytketty samaan päähän lattakaapelin liitintä. Tämä selkeytti sitä, mille välille testattava kaapeli tulisi kytkeä. Lisäksi tämä helpotti vianetsintä tilanteessa vian paikantamisessa. (kuvio 16)



KUVIO 16. Toinen adapterirasian 3D-malli

7 TOTEUTUS

7.1 Ohjelma

Ohjelma kirjoitettiin Arduino IDE-kehitysympäristössä. Arduinolla on oma ohjelmointikielensä, joka perustuu vahvasti C ja C++ ohjelmointikieliin. Ohjelman kirjoittaminen aloitettiin sopivan valikkorakenteen kirjoittamisesta, jonka jälkeen siirryttiin kirjoittamaan siirtorekistereiden ohjaukselle funktio. Kun multipleksereitä saatiin ohjattua siirtorekistereillä, siirryttiin kirjoittamaan kaapelin kytkennän testaavaa funktiota. Tämän jälkeen kirjoitettiin funktiot kaikelle muulle toiminnallisuudelle.

Ohjelmaa kirjoitettaessa pyrittiin hyödyntämään aliohjelmia mahdollisimman tehokkaasti, jottei samaa koodin pätkää tarvinnut kirjoittaa aina uudelleen. Siitä huolimatta lopullisella koodilla oli mittaa yli 1850 riviä. Pääohjelma saatiin kuitenkin todella pieneen tilaan, sillä se koostui vain selkeästä valikkorakenteesta ja aliohjelmien kutsuista.

7.1.1 Pääohjelma

Pääohjelmasta pyrittiin tekemään mahdollisimman selkeä ja lyhyt. Tästä syystä päädyttiin kuvion 17 mukaiseen ratkaisuun, jossa käytetään switch...case -rakennetta. Tämä mahdollistaa erittäin selkeän ja yksinkertaisen ohjelmarungon, jossa jokaisen case-rakenteen alla on vain yksi suurempi aliohjelma, joka taas edelleen koostuu pienemmistä aliohjelmista ja koodista. Pääohjelma koostuu päävalikosta ja asetusvalikosta. Molemmat valikot on toteutettu täysin samalla tavalla, mutta asetusvalikko aukeaa päävalikon sisältä. (Kuvio 17.)

```
void loop()
{
  switch (Menu(5, MainMenu))
  {
    case 1:
      MeasureAndCompare();
      break;
    case 2:
      ReadDataFromSD();
      break;
    case 3:
      MeasureAndSave();
      break;
    case 4:
      FaultyLinesCheck ();
      break;
    case 5:
      switch (Menu(11, SettingsMenu))
      {
        case 1:
          SaveDataToSD();
          break;
        case 2:
          ClockSetup ();
          break;
```

KUVIO 17. Pala pääohjelmasta

7.1.2 Siirtorekistereiden ohjausfunktio

Siirtorekistereiden ohjaamiseen täytyi kirjoittaa funktio. Funktion tarkoituksena oli mahdollistaa siirtorekistereiden lähtöjen ohjaaminen haluttuihin tiloihin sarjamuotoisella signaalilla. Toisin sanoen funktion piti muodostaa 16-bittinen sarjamuotoinen ohjaussignaali, joka saisi siirtorekistereiden lähdöt asettumaan haluttuihin tiloihin multipleksereiden ohjaamiseksi. Molempien sekä syöttävien, että lukevien multipleksereiden ohjaamiseen tarvittiin 4 bittiä. Lisäksi, jotta 16-linjaisilla multipleksereillä voitiin ohjata 48-linjaa, täytyi niitä kytkeä 3 rinnakkain ja päättää enable-signaalilla, minkä multiplekserin halutaan hyödyntävän annettu ohjaus. Tästä syystä molemmille syöttäville ja lukeville multipleksereille tarvittiin vielä 3 bittiä ohjaamaan niiden enable-nastoja. Kokonaisuudessaan tarvittiin siis 14-bittinen ohjaussignaali, mutta koska siirtorekisterit on aina kirjoitettava täyteen, muodostettiin 16-bittinen ohjaussignaali, jossa oli pari tyhjää bittiä.

Muodostettu funktio ottaa syötteenä molemmat sekä kirjoitettavan että luettavan linjan. Linjat annettiin väliltä 1 – 48. Ensimmäiset lukevat ja kirjoittavat multiplekserit ovat vastuussa linjoista 1 – 16, toiset linjoista 16 – 32 ja kolmannet linjoista 32 – 48. Aluksi funktio päättelee ehtolausein annetuista syötteistä, mitkä enable-nastat on ohjattava päälle ja mitkä pois. Sitten nämä 6 bittiä tallennetaan 16-bittisen muuttujan alkuun. Funktio tarkistaa minkälaisia enable-nastoilla valittujen multipleksereiden 4-bittisten ohjaussignaalien on oltava, jotta ne asettuvat haluttuun asentoon. Tämän jälkeen molemmat 4-bittiset ohjaukset tallennetaan samaan edellä mainittuun 16-bittiseen muuttujaan. Lopuksi nämä 16 bittiä lähetetään sarjamuotoisesti mikrokontrollerin digitaalisilla nastoilla siirtorekistereille, jonka seurauksena multiplekserit asettuvat haluttuihin asentoihin.

7.1.3 Testifunktio

Tiedonsiirtokaapeleiden testifunktion tuli lukea mitatun kaapelin kytkentä, verrata sitä ehjän kaapelin kytkentään ja päätellä täsmäsivätkö ne. Funktio ei ota syötteitä. Yksinkertaistetusti funktio toimii seuraavasti. Funktio käyttää siirtorekistereiden ohjausfunktiota ja ohjaa sillä testisignaalin yhteen linjaan kerrallaan. Kun syöttävä multiplekseri syöttää yhtä linjaa niin samalla se lukee testisignaalin kaapelin kaikista nastoista siirtorekistereiden ohjausfunktiota ja lukevia multipleksereitä käyttäen. Tämän jälkeen funktio kääntää testisignaalin syöttösuunnan ja mittaa vielä syöttösuunnan puoleisen lukevan multiplekserin kaikki nasta siirtorekistereiden ohjausfunktiota käyttäen. Sitten funktio siirtyy syöttämään seuraavaa linjaa ja toistaa samaa prosessia, kunnes kaikki linjat on käyty läpi.

Yllä mainitun mittausprosessin aikana mittatulokset käsitellään seuraavasti. Kun testisignaali luetaan mikrokontrollerin A/D-muuntimella niin saatua arvoa verrataan vertailuarvoon, joka on asetettu hieman ehjästä kaapelista mitattua arvoa pienemmäksi. Jos mitattu arvo ylittää vertailuarvon todetaan, että kyseiseen linjaan saatiin yhteys ja tulos merkitään muistiin muuttujaan. Mikäli mitattu arvo alittaa vertailuarvon tarkistetaan, tuleeko nastaan minkäänlaista yhteyttä vertaamalla tulosta arvoon, joka kuvastaa erittäin huonoa yhteyttä. Mitatun arvon ollessa alhaisempi kuin huonoa yhteyttä kuvaava arvo, todetaan ettei yhteyttä ole.

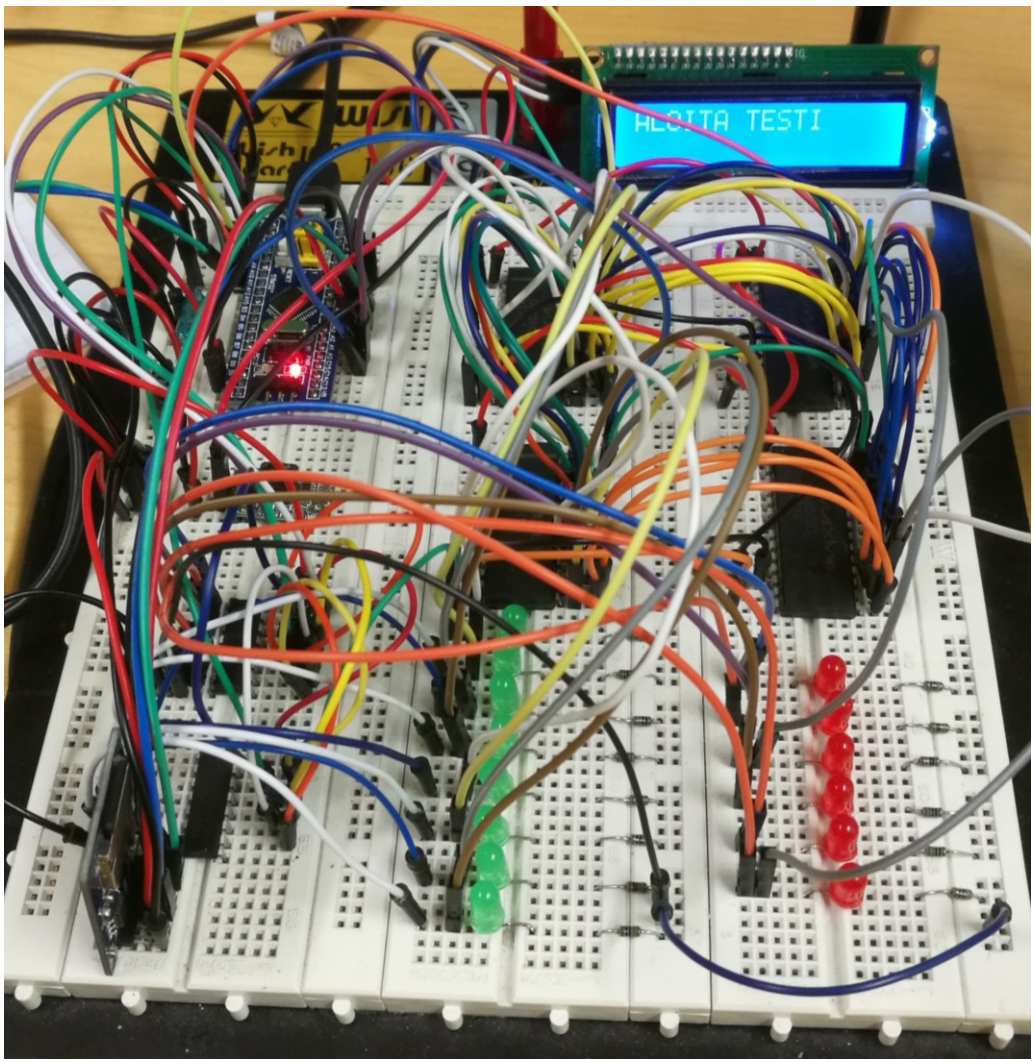
Siinä tapauksessa, että mittatulos kuitenkin ylittää huonoa yhteyttä kuvaavan arvon, mutta alittaa vertailuarvon tarkistetaan, että onko tähän nastaan yhteys ehjässä kaapelissa. Siinä tapauksessa, että yhteys löytyy myös ehjästä kaapelista, todetaan yhteyden olevan heikko eli kyseessä on vika. Mikäli tähän nastaan ei ole yhteyttä ehjässä kaapelissa kyseessä on viallinen yhteys ja se tallennetaan muuttujaan yhteytenä.

Lopuksi tulosmuuttujaa verrataan ehjän kaapelin muuttujaan ja todetaan täsmäävätkö ne. Ehjän ja testattavan kaapelin tulosmuuttujien täsmätessä, kaapeli todetaan ehjäksi, muutoin vialliseksi.

7.2 Prototyyppi

Järjestelmän prototyyppi rakennettiin koekytkentäalustalle. Prototyypin rakentaminen ja ohjelman kirjoittaminen tapahtuivat samaan aikaan. Aina kun osakokonaisuus ohjelmaa oli kirjoitettu, sen toiminta testattiin lataamalla se mikrokontrolleriin ja testaamalla, että ohjelma käyttäytyy halutulla tavalla. Lopullista prototyyppiä ei saatu toteutettua täydessä mittakaavassa niin, että se olisi kyennyt mittaamaan 48-linjaisia tiedonsiirtokaapeleita. Tämä johtui siitä, että koekytkentäalustan koko ei yksinkertaisesti riittänyt. Tästä syystä päädyttiin tekemään prototyyppi, jossa oli vain kolmasosa multipleksereistä, eli sillä pystyttiin mittaamaan vain 16-linjaisia tiedonsiirtokaapeleita. 16-linjaa oli kuitenkin enemmän kuin riittävä määrä kytkennän toimivuuden kokeilemiseen ja tuotteen paranteluun. (kuva 4.)

Siirtorekistereiden lähtöihin kytkettiin ledit, jotta rekistereiden antamaa ohjausta pystyttiin tarkastelemaan prototyypissä. Siirtorekistereinä käytettiin kahta kahdeksan bittistä rekisteriä, joiden mahdollistama ohjaus oli kokonaisuudessaan 16 bittiä. Lopuksi kuitenkin vain 14-bittinen ohjaus riitti haluttuun toiminnallisuuteen. Tästä 14-bittisestä ohjauksesta kahdeksaa bittiä käytettiin ohjaamaan syöttävien sekä lukevien multipleksereiden sisäisten kytkimien asennot ja kuudella bitillä valittiin mitkä multiplekserit ovat päällä.



KUVA 4. Lopullinen prototyyppi koekytkentäalustalla

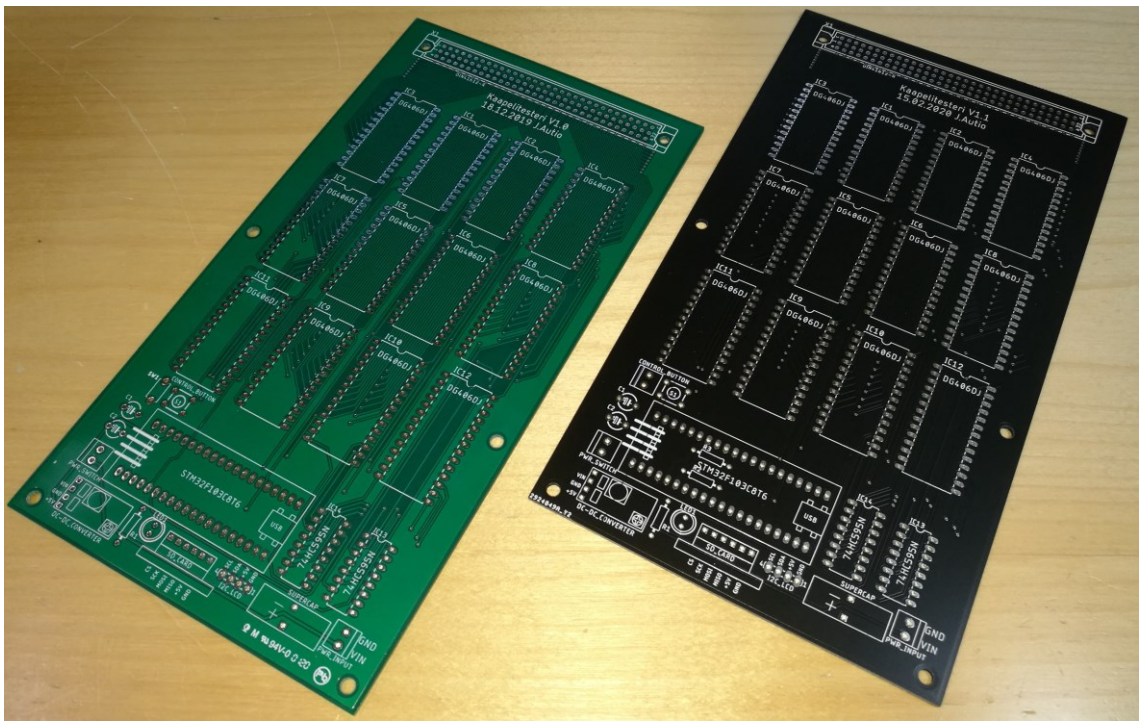
7.3 Piirilevy

Kun laitteen piirilevysuunnitelma oli valmiina, siitä täytyi teetättää piirilevy. Tätä varten piirilevysuunnitelmasta luotiin Autodesk Eagle-ohjelmistossa gerber-tiedosto. Tämä tiedosto lähetettiin piirilevyn tilauksen yhteydessä piirilevyvalmistajalle ja sen pohjalta piirilevyvalmistaja tuotti halutun määrän kyseisiä piirilevyjä. Tilauksen yhteydessä täytyi myös määritellä muun muassa millainen pinnoituksen, värin ja levyn paksuuden tulisi olla.

Piirilevyjä päädyttiin tekemään kaksi eri versiota, sillä ensimmäisestä piirilevy versiosta löytyi testaamisen yhteydessä muutamia pieniä puutteita, joskaan ei mitään toiminnan kannalta ratkaisevaa. Ensimmäinen piirilevy versio tilattiin suo-

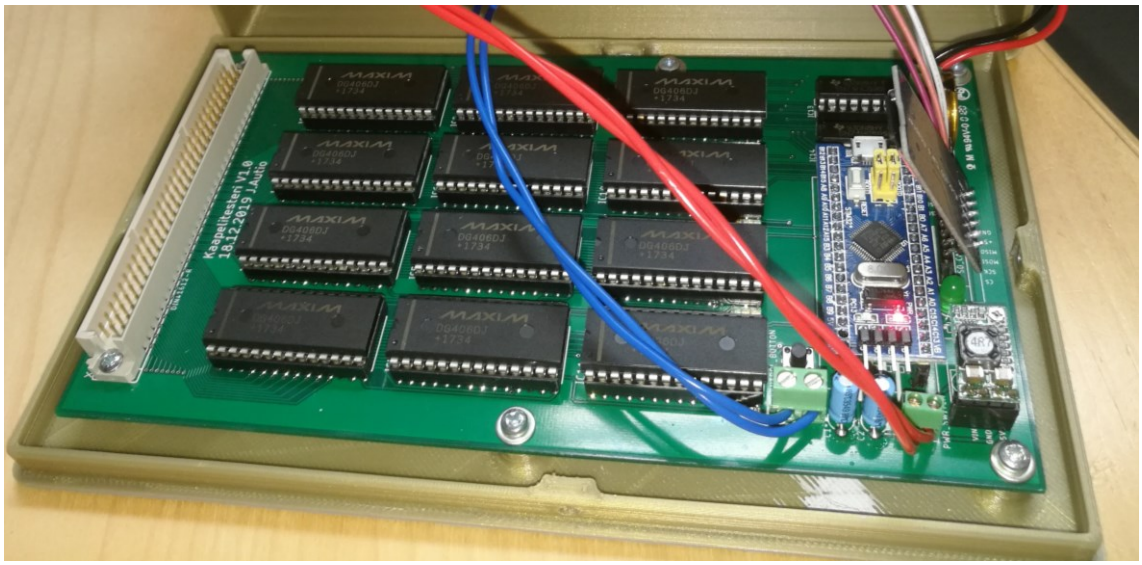
malaiselta piirilevyvalmistajalta nimeltä PCB Connect Oy, joka on Pirkkalasta kotoisin oleva vuonna 2007 perustettu osakeyhtiö. Toimitetussa piirilevyssä itsessään ei ollut mitään vikaa ja laatu oli loistavaa, mutta myöhemmin valmista laitetta testattaessa huomattiin, että testerin tarkkuutta pystyttiin parantamaan korvaamalla mikrokontrollerin sisäiset alasvetovastukset ulkoisilla vastuksilla, jotta niiden arvoja pystyttiin muuttamaan ja tätä kautta optimoimaan järjestelmän mittaustarkkuus. (kuva 5.)

Näiden pienten muokkausten jälkeen päätettiin teetättää myös uudempi piirilevy. Tällä kertaa piirilevy päädyttiin tilaamaan kokeilumielellä kiinalaiselta JLCPCB piirilevyvalmistajalta, sillä hinta oli vain murto-osan suomalaisen valmistajan hinnasta. Lisäksi tilaaminen oli tehty todella helpoksi ja yksinkertaiseksi. Gerber-tiedosto ladattiin suoraan valmistajan sivuille. Tämän jälkeen valittiin tilattavien levyjen määrä ja muut ominaisuudet helposta valikosta. Sitten asetettiin tilaus ja maksettiin verkon välityksellä. Nämä piirilevyt saapuivat noin kahdessa viikossa Kiinasta Suomeen. Piirilevyissä ei laadullisesti ollut minkäänlaista havaittavaa eroa, joka on uskomatonta suureen hintaeroon nähden. Kuvassa 5 vihreä piirilevy on vanhempi versio ja musta uudempi versio.



KUVA 5. Valmiit piirilevyversiot V1.0 ja V1.1

Jotta piirilevy saatiin käyttövalmiiksi, se täytyi vielä kalustaa. Piirilevyt olivat niin isoja ja niihin tuli sen verran paljon komponentteja, että niiden kokoonpanoon kului runsaasti aikaa. Kokoonpano alkoi komponenttien keräämisestä. Kun komponentit oli kerätty, ne juotettiin piirilevylle. Piirejä varten juotettiin kannat, joihin piirit pystyttiin yksinkertaisesti painamaan. Tämä helpotti mahdollisesti viallisten piirien vaihtamista. Myös mikrokontrollerialustaa varten juotettiin liitin, mikä helpotti sen pois ottamista ja ohjelmoimista. Piirilevyn kokoonpanossa pyrittiin sellaiseen ratkaisuun, että piirilevy olisi mahdollisimman modulaarinen komponenttien vaihdon kannalta. (kuva 6.)



KUVA 6. Kalustettu piirilevy keskusyksikössä

7.4 3D- tulostus

Järjestelmän kotelot toteutettiin 3D-tulostamalla. Tulostamiseen käytettiin PLA-muovia ja tulostaminen tehtiin kahdella eri harrastekäyttöön tarkoitetulla 3D-tulostimella. Keskusyksikön kotelon tulostamiseen käytettiin Anet E12 3D-tulostinta. Adapterirasiat puolestaan tulostettiin Anycubic I3 Mega 3D-tulostimella. Tulostamista varten suunnitellut 3D-mallit muutettiin Autodesk Inventor- ohjelmistossa STL-tiedostomuotoon ja avattiin Ultimaker Cura -ohjelmassa, jota käytetään 3D-mallin pilkkomiseen g-koodiksi. FDM-tyyppinen 3D-tulostus tapahtuu kerros kerrokselta, tästä syystä malli on pilkottava erillisellä ohjelmalla ensin g-koodiksi. Tämän jälkeen g-koodia käytetään ohjaamaan 3D-tulostinta. G-koodi

pitää sisällään kaiken 3D-tulostimen tarvitsevan tiedon mallin tulostamista varten kerros kerrokselta.

7.5 Keskusyksikkö

Kun keskusyksikön kansi ja pohja oli 3D-tulostettu, oli aika kasata keskusyksikkö. Järjestelmän keskusyksikkö kasattiin kiinnittämällä sen koteloon kaikki ohjauselektroniikka kuten painonappi, virtakytkin, DC-liitin ja LCD-näyttö. Keskusyksikön pohjalevyyn tulleet mutterit liimattiin niille luotuihin upotuksiin, ja piirilevyn kiinnitystä varten tarkoitetut kierreholkit sulatettiin pohjalevyssä oleviin paikkoihin käyttäen juotinta. Sitten piirilevy ruuvattiin kiinni ja tämän jälkeen kaikki kanteen kiinnitetyt osat johdotettiin piirilevylle. Kuvassa 7 on esitettynä valmis keskusyksikkö.

KUVA 7. Valmis keskusyksikkö

7.6 Adapterirasiat

Kun adapterirasioiden kotelot oli saatu 3D-tulostettua, oli aika kasata ne. Aluksi tiedonsiirtokaapeleiden liittimien nastoihin juotettiin riittävän pitkät johtimet, jotta ne voitiin kytkeä lattakaapelin liittimeen. Tämän jälkeen liittimet ruuvattiin niille tehtyihin upotuksiin. Lopuksi johtimien vapaat päät juotettiin järjestyksessä lattakaapeliliittimen molempiin päihin. (kuva 8)

KUVA 8. Adapterirasian kytkentä

Ensimmäisellä adapterirasialla pystyttiin testaamaan neljää erilaista tiedonsiirtokaapelia, joista kolmessa oli käytössä samat liittimet, mutta kaikilla näistä kaapeleista oli silti eroavat sisäiset kytkennät. Jokaiseen adapterirasiaan käytettiin samanlaista 96-nastaista liitintä lattakaapelille. Tämä 96-nastainen liitin mahdollisti 48-linjasen kaapelin kytkemisen adapterirasiaan sen molemmista päistä. Liittimiä yhteen adapterirasiaan voitiin kuitenkin laittaa niin monta kuin sen koteloon saatiin mahtumaan. Tämä johtuu siitä, että liittimet voidaan johdottaa lattakaapelin

liittimeen rinnakkain kuten kuvasta 8 nähdään. Ensimmäisestä adapterirasiasta tuli erittäin siisti ja liittimet istuivat hyvin sen kanteen kuten kuvasta 9 nähdään.

KUVA 9. Valmis adapterirasia

Toinen toteutettu adapterirasia oli suunniteltu huomattavasti isommille liittimille ja liittimiä tuli siihen useampia. Kuten ensimmäiselläkin adapterirasialla, myös sillä voitiin testata neljää erilaista tiedonsiirtokaapelia.

7.7 Valmis järjestelmä

Kokonaisuudessaan tiedonsiirtokaapeleiden testausjärjestelmä koostuu keskusyksiköstä ja tarvittavasta määrästä siihen lattakaapelilla kytkettäviä adapterirasioita. Tämän opinnäytetyön aikana ehdittiin toteuttaa kaksi adapterirasiaa, joista

molemmat mahdollistivat neljän erilaisen tiedonsiirtokaapelin kytkemisen testausjärjestelmän keskusyksikköön. Järjestelmästä tuli kompakti ja erilaiset adapterirasiat oli helppo kytkeä keskusyksikköön lattakaapelilla. Lattakaapeli mitoitettiin riittävän pitkäksi, jotta kytkettävät adapterirasiat saatiin riittävän kauas keskusyksiköstä. Tämä tehtiin siitä syystä, ettei kaapeleita kytkettäessä, vahingossa vaurioiteta keskusyksikköä. Järjestelmään on helppoa lisätä tarpeen vaatiessa uusia adapterirasioita ja tällä tavoin sen testauskapasiteettia saadaan kasvatettua entisestään. Kuvassa 10 esitettynä valmis tiedonsiirtokaapeleiden testausjärjestelmä työpöydällä.

KUVA 10. Valmis järjestelmä

8 JÄRJESTELMÄN TESTAAMINEN

Kun järjestelmä oli saatu valmiiksi, haluttiin sen ominaisuuksia ja toimintavarmuutta testata. Aluksi järjestelmää testattiin kytkemällä testerin lattakaapeliin oikosulku hyppylangalla. Tämän jälkeen testausjärjestelmälle opetettiin kyseinen kytkentä siihen tarkoitetulla funktiolla, minkä jälkeen suoritettiin testifunktio kytkennän testaamiseksi. Tässä tilanteessa järjestelmän tuli todeta kytkentä oikeelliseksi. Testifunktiota toistettiin 5 kertaa, jonka jälkeen oikosulku poistettiin liittimestä. Sitten testifunktiota toistettiin jälleen 5 kertaa. Tällä kertaa testerin tuli todeta kytkentä aina vialliseksi.

Tämän testin aikana havaittiin satunnaisia virheellisiä testejä. Vaikutti siltä, että kyseessä oli häiriöluontoinen vika. Hetken vianetsinnän jälkeen havaittiin vian todellakin olevan häiriölähtöistä. Todettiin, että suojaamaton lattakaapeli toimii hieman kuin antenni ja keräsi häiriösignaalia itseensä. Lisäksi todettiin, että mikrokontrollerin A/D- muuntimessa oleva kondensaattori ei ehtinyt asettua luettavan nastan jännitteeseen.

Koska häiriösuojauksen toteuttaminen koteloihin jälkikäteen olisi ollut erittäin haastavaa, pyrittiin ongelma ratkaisemaan ohjelmallisesti. Ongelma ratkaistiinkin kirjoittamalla A/D-muuntimen lukemiseen erillinen funktio, joka luki arvon useaan kertaan ja vertasi luettua arvoa aina edelliseen luettuun arvoon. Kun sama tai lähes sama arvo saatiin riittävän monta kertaa putkeen, voitiin todeta tulos oikeelliseksi. Tämän jälkeen toimintaa testattiin uudestaan ja todettiin ongelman poistuneen.

Kun perustavanlaatuinen testi oli suoritettu, haluttiin testata järjestelmän tarkkuutta. Tarkkuutta testattiin samaan tapaan kuin ensimmäisessä testissä, mutta tällä kertaa hyppylangalla aiheutetun oikosulun poistamisen sijaan se korvattiin vikaresistanssia kuvaavalla potentiometrillä. Tämän jälkeen testausjärjestelmällä suoritettiin testifunktioita eri vastusarvoilla ja näin tarkastettiin, kuinka suurella vikaresistanssilla järjestelmä toteaa linjan olevan poikki. Lopputuloksena päästiin siihen tulokseen, että noin 10 ohmin vikaresistanssi lisääminen linjaan riitti katkoksen toteamiseen.

Kun edellä mainitut testit oli suoritettu, voitiin siirtyä oikeiden tiedonsiirtokaapeleiden testaamiseen. Järjestelmään kytkettiin adapterirasia ja tiedonsiirtokaapeli. Ensiksi tiedonsiirtokaapelin kytkentä opetettiin järjestelmälle siihen tarkoitetulla funktiolla, ja sitten suoritettiin testifunktio kaapelin testaamiseksi. Tämän jälkeen testausjärjestelmän tuli todeta kaapeli ehjäksi jokaisella testikerralla. Testejä suoritettiin 10, jotta voitiin olla varmoja, että järjestelmään voidaan luottaa. Kun testi saatiin menemään 10 kertaa läpi, aiheutettiin adapterin rinnakkaiseen liittimeen oikosulku hyppylangalla ja suoritettiin testifunktio jälleen. Tällä kertaa järjestelmä totesi kaapelissa olevan vikaa, mikä tarkoitti sitä, että järjestelmä toimi halutulla tavalla.

Myös järjestelmän helppokäyttöisyyttä haluttiin kokeilla. Todettiin, että paras tapa tähän oli pyytää asentaja, jolla ei ollut minkäänlaista aikaisempaa kokemusta testausjärjestelmän käytöstä, kokeilemaan järjestelmää. Tätä varten järjestelmälle oli opetettu kolmen kaapelin kytkennät ja kyseiset kolme kaapelia asetettiin järjestelmän kanssa pöydälle. Tarkoitus oli kokeilla, kuinka nopeasti asentaja kykenee oppimaan testerin käytön ja testaamaan nuo kolme kaapelia. Tuloksena tästä testistä selvisi, että pienen sanallisen ohjeistuksen kanssa jokainen asentaja pystyi oppimaan järjestelmän käytön alle viidessä minuutissa, mikä oli upeaa huomata. Näiden testien koettiin olevan riittävä peruste järjestelmän käyttöönottoon; usein viat ja puutteet havaitaan parhaiten käyttämällä järjestelmää oikeissa tilanteissa ja ajan kanssa.

9 POHDINTA

Opinnäytetyön tarkoituksena oli kehittää monilinjaisten tiedonsiirtokaapeleiden testausjärjestelmä, jolla pystytään nopeuttamaan ja parantamaan nykyistä tiedonsiirtokaapeleiden testausprosessia. Tavoitteena oli saada järjestelmästä luotettava ja toimintavarma, lisäksi pyrittiin saamaan siitä niin helppokäyttöinen kuin mahdollista. Järjestelmä piti kehittää ideasta aina valmiiksi konkreettiseksi laitteeksi asti. Tämän takia työ jakaantuikin seuraavanlaisiin osa-alueisiin: Toimintaperiaate, komponenttivalinnat, ohjelmointi, prototyyppi, piirilevy, 3D-mallinnus ja 3D-tulostus. Kyseessä oli siis hyvin laaja ja monipuolinen kokonaisuus.

Koska kyseessä oli tuotekehitysprojekti, sen toteuttaminen itsenäisesti vaati erittäin monipuolista osaamista ja jatkuvaa uuden opettelemista. Yleensä tuotekehitystä varten luodaan projektiryhmä useiden eri osa-alueiden asiantuntijoista. Tällöin jokainen ryhmän jäsen on vastuussa vain omasta ydinosaamisestaan. Tästä syystä tämä projekti olikin hyvin haastava toteuttaa itsenäisesti, sillä työssä oli keskityttävä jokaiseen osa-alueeseen ja samalla myös kokonaisuuteen. Toisaalta tämä teki työstä myös erittäin opettavaisen ja mielenkiintoisen. Näin työstä tuli myös paljon henkilökohtaisempi, sillä koko kokonaisuus oli oman työn tulosta. Lopuksi kun työssä päästiin kunnolla vauhtiin, niin se tuntui jo enemmän harrastukselta kuin työltä.

Tuloksena tästä työstä syntyi täysin toimiva monilinjaisten tiedonsiirtokaapeleiden testausjärjestelmä. Järjestelmästä tuli erittäin monipuolinen ja siihen saatiin sisällytettyä paljon toiminnallisuutta. Lopullinen järjestelmä saatiin kehitettyä jopa niin pitkälle, että siihen pystyttiin lisäämään ominaisuuksia, jota teknisessä määrittelyssä ei ollut edes esitetty. Testausjärjestelmä täytti kaikki sille asetetut vaatimukset selkeällä marginaalilla ja osoittautui kokonaisuudessaan varsin toimivaksi ratkaisuksi. Tämä oli hienoa huomata, sillä työhön käytettiin erittäin paljon aikaa ja työkuorma oli ajoittain todella raskas.

Kokemusta järjestelmän käytöstä pidemmällä aikavälillä ei vielä ole. Tästä syystä uskon, että paranneltavaa ja kehitettävää tulee vielä löytymään, sillä to-

dennäköistesti järjestelmän vahvuudet ja heikkoudet tulevat esiin vasta oikeassa käytössä. Toistaiseksi kuitenkin kaikki kohdatut ongelmat sekä vastoinkäymiset saatiin voitettua ja tuote toimii halutulla tavalla. Tällä hetkellä järjestelmän tarkkuus on riittävä sen käyttötarkoitukseen, mutta mikrokontrolleria vaihtamalla ja pienellä ohjelmamuutoksella olisi mahdollista parantaa sen tarkkuutta entisestään. Jää kuitenkin nähtäväksi tullaanko tällaista päivitystä koskaan toteuttamaan, sillä siihen ei ainakaan vielä ole tarvetta. Toistaiseksi järjestelmää testanneiden palaute on ollut ainoastaan positiivista, eikä käytössä ole ilmennyt ongelmia. Myös järjestelmän helppokäyttöisyydestä ja käytön yksinkertaisuudesta on pidetty.

LÄHTEET

Afzal, S. Analog Devices. 2020. ”I2C Primer: What is I2C? (Part 1)”. [www-sivu]. Luettu 27.3.2020. https://www.analog.com/en/technical-articles/i2c-primer-what-is-i2c-part-1.html

Arduino. 2020. Arduino kotisivut. [www-sivu]. Luettu 22.1.2020. http://arduino.cc/en/Reference/HomePage

CircuitsToday. 2020. ”Liquid Crystal Displays (LCD) – Working”. [www-sivu]. http://www.circuitstoday.com/liquid-crystal-displays-lcd-working

Farnell. ”Arduino Nano”. Datalehti. http://www.farnell.com/datasheets/1682238.pdf

Joy-it. ”I2C Serial 2.6“ LCD Module”. Datalehti. https://www.partco.fi/fi/index.php?controller=attachment&id_attachment=1747

Learning electronics. 2003. " Serial-in, parallel-out shift register" [www-sivu]. Luettu 8.3.2020. http://www.learningelectronics.net/vol_4/chpt_12/4.html

Millog Oy. 2018. Millog kotisivut. [www-sivu]. Luettu 12.3.2020. https://millog.fi/yritys/

Postolache, O., Silva Girão, P. & Dias Pereira, J.M. 2010. Non-Volatile Memory Interface Protocols for Smart Sensor Networks and Mobile Devices. Instituto de Telecomunicações Portugal. 10.5772/8864. Luettu 20.2.2020. https://www.researchgate.net/publication/221908590_Non-Volatile_Memory_Interface_Protocols_for_Smart_Sensor_Networks_and_Mobile_Devices#pf10

STMicroelectronics. ”Medium-density performance line ARM®-based 32-bit MCU with 64 or 128 KB Flash, USB, CAN, 7 timers, 2 ADCs, 9 com. interfaces”. DocID13587 Rev 17. Datalehti. https://www.st.com/resource/en/datasheet/stm32f103c8.pdf

Suomen asiakastieto Oy. 2020. Millog Oy. Luettu 25.2.2020. https://www.asiakastieto.fi/yritykset/fi/millog-oy/20518595/yritys

Techshopbd. 2020. ”STM32F103C8T6 Blue pill Arduino guide”. Luettu 1.3.2020. https://www.techshopbd.com/uploads/product_document/STM32bluepillarduinoguide(1).pdf

Vishay Siliconix. ”16-Ch/Dual 8-Ch High-Performance CMOS Analog Multiplexers”. Datalehti. https://www.vishay.com/docs/70061/dg406.pdf